SMC-TR-02-34

AEROSPACE REPORT NO.
TR-2002(8550)-3

A Framework for Software Products Within a System Context (2nd Edition)

31 May 2002

Prepared by

R. J. ADAMS
Software Systems Acquisition Department
Computer Systems Division

S. ESLINGER
Software Engineering Subdivision
Computer Systems Division

Prepared for

SPACE AND MISSILE SYSTEMS CENTER
AIR FORCE SPACE COMMAND
2430 E. El Segundo Boulevard
Los Angeles Air Force Base, CA 90245

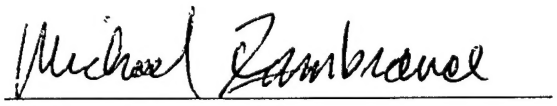
Engineering and Technology Group

THE AEROSPACE CORPORATION
El Segundo, California

This report was submitted by The Aerospace Corporation, El Segundo, CA 90245-4691, under Contract No. F04701-00-C-0009 with the Space and Missile Systems Center, 2430 E. El Segundo Blvd., Los Angeles Air Force Base, CA 90245. It was reviewed and approved for The Aerospace Corporation by Mary A. Rich, Principal Director, Software Engineering Subdivision. Michael Zambrana was the project officer for the Mission-Oriented Investigation and Experimentation (MOIE) program.

This report has been reviewed by the Public Affairs Office (PAS) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nationals.

This technical report has been reviewed and is approved for publication. Publication of this report does not constitute Air Force approval of the report's findings or conclusions. It is published only for the exchange and stimulation of ideas.



Michael Zambrana
SMC/AXE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE (*DD-MM-YYYY*)	2. REPORT TYPE	3. DATES COVERED *(From - To)*
31-05-2002		

4. TITLE AND SUBTITLE
A Framework for Software Products Within a System Context (2nd Edition)

5a. CONTRACT NUMBER
F04701-00-C-0009

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Richard J. Adams
Suellen Eslinger

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

The Aerospace Corporation
Laboratory Operations
El Segundo, CA 90245-4691

8. PERFORMING ORGANIZATION REPORT NUMBER

TR-2002(8550)-3

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Space and Missile Systems Center
Air Force Space Command
2450 E. El Segundo Blvd.
Los Angeles Air Force Base, CA 90245

10. SPONSOR/MONITOR'S ACRONYM(S)
SMC

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
SMC-TR-02-34

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Any software development standard and all software products required by that standard need to be understood within the context of the total set of related system standards and products if they are to be appropriately applied to a program development. For SMC/NRO programs, software seldom, if ever, stands alone independent of a larger hardware/software system context. This report provides the context to help the SMC/NRO SPOs or contractors with the task of defining software and software-related system-level products and their contents.

This report describes the consistent philosophical structure needed to develop the tailoring of the software products defined by EIA/IEEE J-STD-016-1995 and their associated software-related system-level products. The EIA/IEEE standard is the commercial standard based on MIL-STD-498. This report documents the assumptions about the software products and their information content and defines the relationship of these software products with each other and with any software-related system-level products. System or software products that are needed but are not currently defined by any standard have been identified, and their content and the relationship to the existing software or system products has been defined.

The figures and descriptions contained in this report do not imply that a specific physical document is required. The actual instantiation of any product described in this report is totally at the discretion of a particular SMC/NRO SPO or contractor. This report only intends to convey the class of information that must be available in some meaningful form and the relationships among the information.

This report is divided into three sections that cover all of the system products. The three sections are "Plan Relationships," "Requirements/Design/Verification Relationships," and "System Operations and Maintenance Relationships." Some products appear in multiple sections since, as they are currently structured, they perform multiple functions in the context of the total set of system products.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Suellen Eslinger
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED		79	19b. TELEPHONE NUMBER *(include area code)* (310) 336-2906

Executive Summary

1. Introduction

Any software development standard and all software products required by that standard need to be understood within the context of the total set of related system standards and products if they are to be appropriately applied to a program development. For SMC/NRO programs, software seldom, if ever, stands alone independent of a larger hardware/software system context. The task of developing the set of tailoring for all of the system standards and products is beyond the scope and intent of this report. However, this report provides the context to help the SMC/NRO SPOs or contractors with the task of defining the software and software-related system-level products and their contents.

1.1 Scope

This report describes the consistent philosophical structure needed to develop the tailoring of the software products defined by EIA/IEEE J-STD-016-1995[1] and its software-related system-level products. It documents the assumptions about the software products and their information content, and defines the relationship of these software products with each other and with any software-related system-level products.

1.2 Organization and Content

This report identifies the total set of system products that will need tailoring by the SMC/NRO SPOs or contractors. It specifically identifies the software products and defines the relationships between the software products, defined by EIA/IEEE J-STD-016-1995, and the other necessary system products.

The figures and descriptions contained in this report do not imply that a specific physical document is required. The actual instantiation of any product described in this report is totally at the discretion of a particular SMC/NRO SPO or contractor. This report only intends to convey the class of information that must be available in some meaningful form and the relationships among the information.

This report is divided into three sections that will cover all of the system products. The three sections are "2. Plan Relationships," "3. Requirements/Design/Verification Relationships," and "4. System Operations and Maintenance Relationships." Some products appear in multiple sections since, as they are currently structured, they perform multiple functions in the context of the total set of system products.

Each section contains a primary figure showing the total set of products associated with that section and the relationships between the products. This report assumes a program of sufficient complexity to need the entire hierarchy of system products. The legend in each primary figure specifically identifies those software products contained in EIA/IEEE J-STD-016-1995. The legend also identifies those software-related system products that have a suitable existing Data Item Description (DID) that can be used as the basis for tailoring and those system products that do not.

Each primary figure uses the following conventions. Major related elements are enclosed in heavy black boxes with a title. Elements within boxes (or sometimes between boxes) that have a strong peer-to-peer relationship are joined with a single heavy line. This line does not imply a one-to-one

[1] EIA/IEEE J-STD-016-1995 is the commercial standard based on MIL-STD-498.

relationship. Peer-to-peer relationships can be one-to-many or many-to-many as well. Elements within boxes that have a strong hierarchical relationship are joined by double-headed arrows in the shape of an "X." Relationships between major elements enclosed within the heavy box are shown with large double-headed arrows.

2. Plan Relationships

Figure 1 describes the relationships between management plans. It assumes that the "Acquisition Organization" develops an acquisition plan that will be used as a management agreement between the "Acquisition Organization" and its management and between the "Acquisition Organization" and the "User Organization" for the acquisition of the system to be developed. This agreement and those documents referenced by the agreement (e.g., the Acquisition Program Baseline), define the basic philosophy and approach for the acquisition of the system and define the fundamental characteristics the system must meet if the development is to proceed. For the purposes of this report, this agreement is assumed to be documented in the Single Acquisition Management Plan (SAMP) and the other documents are assumed to be included in the SAMP by reference. It assumes that the "Acquisition Organization" develops a master verification plan to ensure that the system meets its requirements prior to acceptance by the "Acquisition Organization." For the purposes of this report, this master verification[2] plan is documented in the Test and Evaluation Master Plan (TEMP).

The planning products for the system are shown in three boxes. Within the "Formal Agreement Plan" box is the Integrated Master Plan (IMP). This plan, including its associated narrative explanation, is developed by the contractor and, in the case of the prime contractor, it is included as part of the development contract when the contract is awarded. This plan contains program characteristics that govern all other planning for the system.

Within the "System Plans" box, plans and schedules are shown as aggregates under the primary topics that must be covered for system acquisition. These aggregate plans and schedules encompass the entire system life cycle and form the basis for lower-level software and hardware development and transition[3] planning and scheduling.

Within the "Development and Transition Plans" box, the software development and software transition plans and schedules encompass the entire computer resources[4] life cycle from system requirements definition through turnover to operations and maintenance (O&M), and include planning and scheduling for the use of computer resources during software, hardware and system verification and transition.

---

2 The term "verification" rather than "test" has been used consistently throughout to emphasize the fact that the system and its components are verified rather than tested. The term "test" is only used in the context of one of the four verification methods, namely, Inspection, Analysis, Demonstration and Test.

3 Transition planning is the detailed planning to transfer a product from one organization to another.

4 The term "computer resources" rather than "software" has been used consistently throughout this report to emphasize the fact that software never stands alone outside the context of the hardware system on which it operates. This is true even when the hardware is a commercial product.

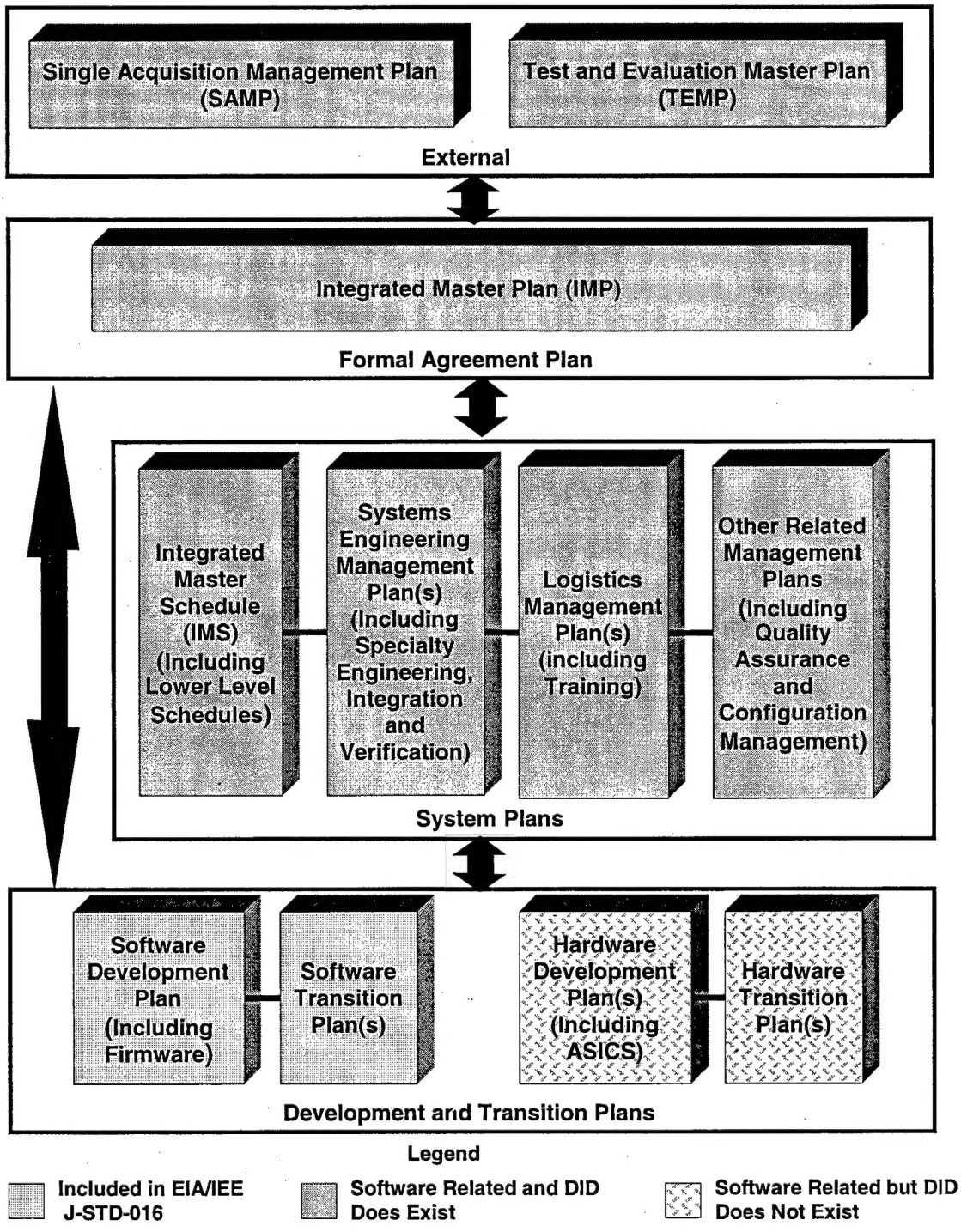


Figure 1. Plan relationships.

3. Requirements/Design/Verification Relationships

Figure 2 describes the requirements, design and verification product relationships. It assumes that the "User Organization" develops their own system requirements and concept of operations and documents them in some formal document. For the purposes of this report, the user's requirements

are documented in the Operational Requirements Document (ORD). It assumes that the "Acquisition Organization" develops an abbreviated set of encompassing performance-related requirements and documents them in some formal document. For the purposes of this report, the acquisition organization's requirements are documented in the System Performance Document (SPD). It further assumes that a "Development Organization" creates an acquisition product hierarchy for the system to be acquired. This acquisition product hierarchy consists of: a) a system, segment[5] and B-Level item[6] specification hierarchy containing the system's requirements (known as the specification tree), b) a design product hierarchy containing the system's architectural description and design documents that relate to the specification tree and contains the system's design, and c) a verification product hierarchy that also relates to the specification tree and contains the system's specific verification plans, descriptions and reports.

The requirements, design and verification products for the system are shown in three boxes. Requirements products are shown in the middle box ("Requirements") since both design and verification products flow from requirements products. External requirements shown in the "External Requirements" box are shown for completeness but are not intended to be included as part of the system products needed for future tailoring. Within the "Verification" box, verification plans, verification descriptions and verification reports are shown at each level of the system verification hierarchy.

Within the "Design" box, the System/Segment Design Description (SSDD) and the Operations Concept Description (OCD) provide the mechanism to describe and maintain the total system architecture (including computer resources) and the philosophy of operation for the life of the system (both development and maintenance). The SSDD also provides the primary reference(s) to the requirements, allocation and verification traceability repository. This repository (not shown in the figure) consists of one or more physical information structures that provide system-wide bi-directional traceability for all system requirements (including interface requirements), allocable items (e.g., performance timelines and reliability) and requirements verification records (e.g., verification methods, levels, procedures, and status of each requirement) that are to be maintained for the life of the system.

---

5 The term "segment" rather than "subsystem" (used by EIA/IEEE J-STD-016) has been used consistently throughout this report because "segment" is the more common term for the described entity in the SMC/NRO and associated contractor community.

6 For clarity, the term "B-Level item" as used here encompasses all hardware, combined hardware and software, and software configuration items contained in the specification tree below the level of the segment(s). The term specifically includes Software and Interface Requirement Specifications.

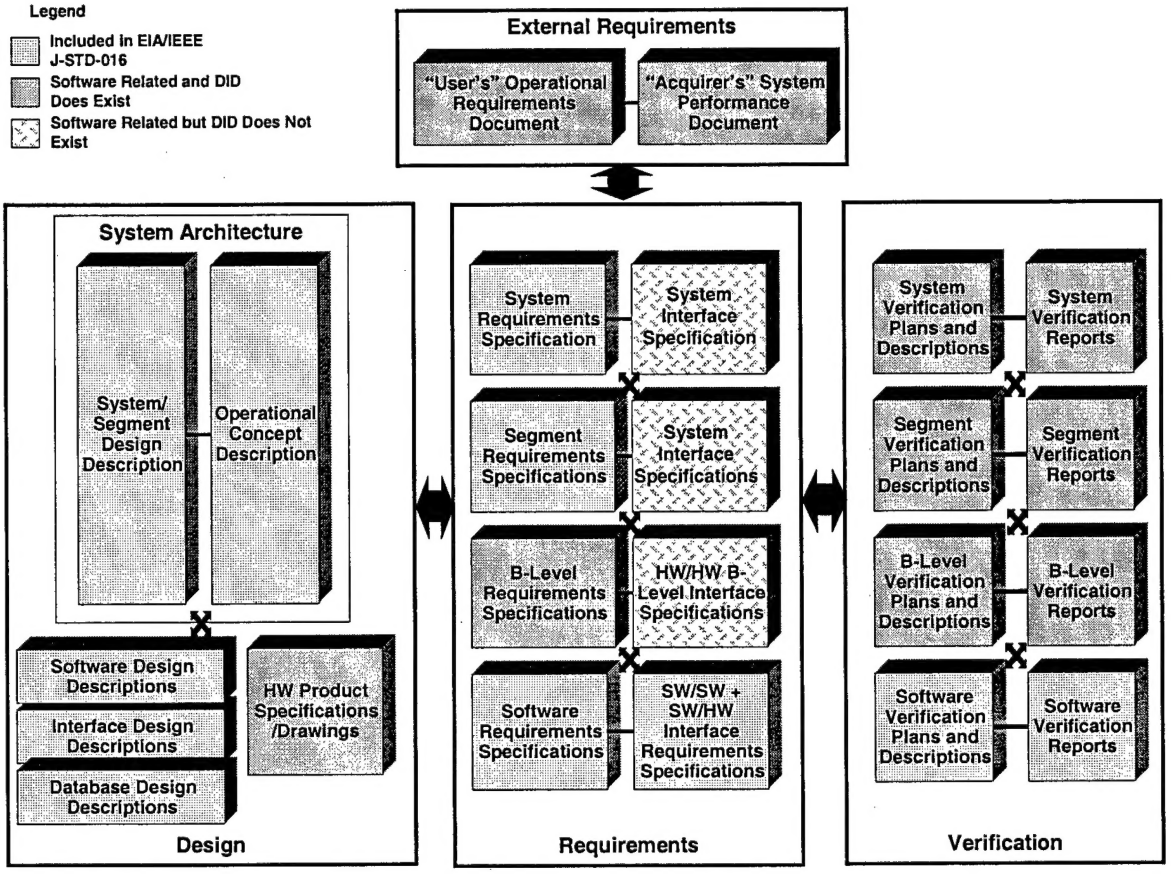


Figure 2. Requirements/design/verification relationships.

4. System Operations and Maintenance Relationships

Figure 3 describes the relationships between system operations and maintenance products.

The "System Operation" box includes all of the operator positions that directly or indirectly support the primary purpose of the system. These operators perform nominal operations related to the scope of their responsibilities in the system as well as certain administrative operations and non-nominal system maintenance operations. The "Design (System Architecture)" box contains the as-built maintenance versions of the documentation shown in the "System Architecture" sub-box in the "Design" box of Figure 2.

The "System Maintenance" box contains the information needed for performing computer resources maintenance. This maintenance is divided into two very distinct functions. The first is maintenance in the sense of performing Fault Detection (FD) and Fault Isolation (FI) in order to restore the computer resources to some operational state (even if it is in some form of degraded mode) as quickly as possible. This form of maintenance is called "System Restoral" and is performed by both operations and maintenance personnel. The term "System Restoral" applies to all system functions, both hardware and software. The second is maintenance in the sense of performing Fault Detection (FD), Fault Isolation (FI) and Fault Localization (FL) in order to repair a fault in the computer resources. This form of maintenance is called "System Repair" and is performed by maintenance personnel. The term "System Repair" applies to all system functions, both hardware and software.

The "System Operation and Maintenance" box contains two types of information. The first is additional information needed to support nominal operations. The second is the complementary system restoral and system repair information needed for the non-computer resources portions of the system.

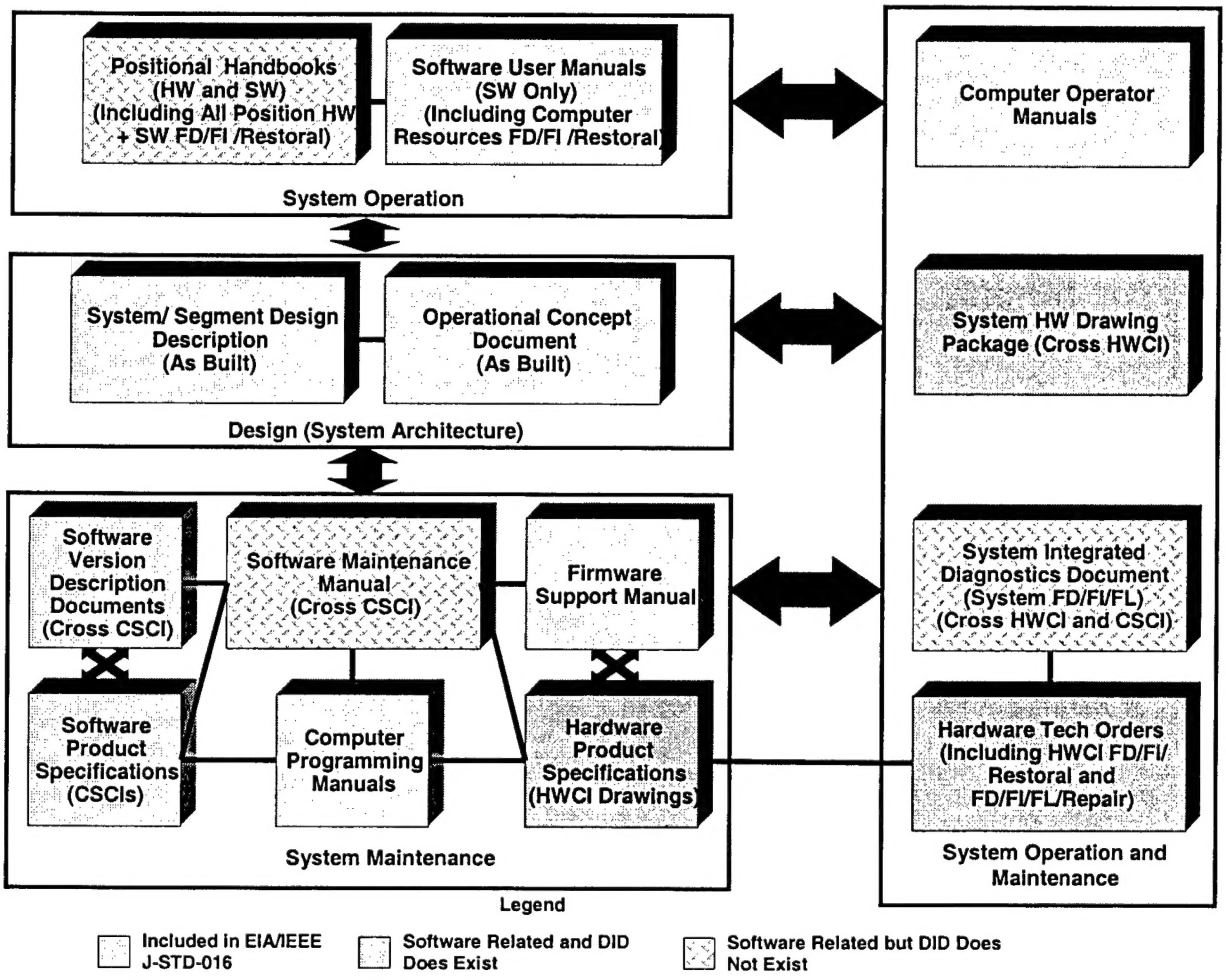


Figure 3. System operations and maintenance relationships.

Contents

Figures

Tables

List of Acronyms

C&C	Command and Control
CIDS	Critical Item (Development) Specification
CM	Configuration Management
COM	Computer Operator Manual
CONOPS	Concept of Operation
COTS	Commercial off-the-Shelf
CSCI	Computer Software Configuration Item
DBDD	Database Design Description
DID	Data Item Description
ECP	Engineering Change Proposal
FD	Fault Detection
FI	Fault Isolation
FL	Fault Localization
FSM	Firmware Support Manual
HDP	Hardware Development Plan
HIS	Hardware Interface Specification
HTrP	Hardware Transition Plan
HWCI	Hardware Configuration Item
ICD	Interface Control Document
IDD	Interface Design Document
IMP	Integrated Master Plan
IMS	Integrated Master Schedule
IPT	Integrated Product Team
IRS	Interface Requirements Specification
LID	Layered Interface Definition
OCD	Operations Concept Description
ORD	Operational Requirements Document
OSIRM	Open Systems Interconnection Reference Model

List of Acronyms (continued)

PIDS	Prime Item (Development) Specification
PROM	Programmable ROM
QA	Quality Assurance
ROM	Read-Only Memory
SAMP	Single Acquisition Management Plan
SDD	Software Design Document
SDP	Software Development Plan
SEMP	Systems Engineering Management Plan
SMC/NRO	Space and Missile Systems Center/National Reconnaissance Office
SPD	System Performance Document
SPO	System Program Office
SPS	Software Product Specification
SRS	Software Requirements Specification
SSDD	System/Segment Design Description
SSIS	System/Segment Interface Specification
SSS	System/Segment Specification
STrP	Software Transition Plan
SUM	Software User Manual
SVD	Software Version Description
TEMP	Test and Evaluation Master Plan

1. Introduction

Any software development standard and all software products required by that standard need to be understood within the context of the total set of related system standards and products if they are to be applied appropriately to a program development. For Space and Missile Systems Center/National Reconnaissance Office (SMC/NRO) programs, software seldom, if ever, stands alone independent of a larger hardware/software system context. In this regard, to be effective, any tailored[1] software development products need an accompanying consistent set of tailored system products. This report provides the context to help the SMC/NRO SPOs or contractors develop the complete set of tailored system and software development products.

1.1 Background

The evaluation of Data Item Description (DID) tailoring efforts performed on multiple earlier programs identified that certain discrepancies existed between the tailoring done by different team members that were more fundamental than just the proposed wording of the tailoring. We discovered that each member of the team had a slightly different perspective on the software development process. This difference in perspective encompassed the content of the software products and their relationships described in the standards. It also encompassed the content of software-related system products and their relationships to each other and to the software products. These subtle differences in outlook and perspective resulting from differences in experience translated into unwritten assumptions that influenced the tailoring of the individual DIDs. This, in turn, produced tailoring that was acceptable within the context of a given software product but inconsistent when taken in the context of the entire set of software products. This discovery convinced the team that it was critical to the success of any tailoring effort that a consistent philosophy be developed and documented. This consistent philosophy would then provide the underlying philosophy and perspective used to tailor each of the software products defined by the standards.

1.2 Scope

This report describes the consistent philosophical structure needed to develop the tailoring of the software products defined by EIA/IEEE J-STD-016-1995[2] (Ref. 2) and its related system-level products. It documents the assumptions about the software products and their information content, and defines the relationship of these software products with each other and with any software-related system-level products. System or software products that are needed but are not defined currently by any standard have been identified, and their content and the relationship to the existing software or system products has been defined.

1.3 Organization and Content

This report identifies the total set of system products that will need tailoring by the SMC/NRO SPOs or contractors. It defines the relationships between the software products, defined by EIA/IEEE J-STD-016-1995, and the other necessary system products.

---

[1] For the purpose of this report, to tailor a set of documents is to change the description of the content of one or more of those documents to ensure that the intent of that set of documents is expressed completely in unambiguous language without contradictions.

[2] This IEE/EIA standard is the successor standard to MIL-STD-498 (Ref 1).

The figures and descriptions contained in this report do not imply that a specific physical document is required. The actual instantiation of any product described in this report is totally at the discretion of a particular SMC/NRO SPO or contractor. This report intends only to convey the class of information that must be available in some meaningful form and the relationships among the information.

This report is divided into three sections that will cover all of the system products. Sections 2 through 4 are "Plan Relationships," "Requirements/Design/Verification Relationships," and "System Operations and Maintenance Relationships." Some products appear in multiple sections since, as they currently are structured, they perform multiple functions in the context of the total set of system products.

Each section contains a primary figure (Figures 1, 13, and 20) showing the total set of products associated with that section and the relationships among those products. The text of each section provides information about the elements of the figure. This report assumes a program of sufficient complexity to need the entire hierarchy of system products. The legend in each primary figure specifically identifies those software products contained in EIA/IEEE J-STD-016-1995. The legend also identifies those software-related system products that have a suitable existing DID that can be used as the basis for tailoring and those system products that do not. When no existing DID is suitable for use as a starting point for tailoring, then for those specific system products, the tailoring task will consist of the creation of an entire one-time DID.

Each primary figure uses the following conventions. Major related elements are enclosed in heavy black boxes with a title. Elements within boxes (or sometimes between boxes) that have a strong peer-to-peer relationship are joined with a single heavy line. This line does not imply a one-to-one relationship. Peer-to-peer relationships can be one-to-many, many-to-one, or many-to-many as well. Elements within boxes that have a strong hierarchical relationship are joined by double-headed arrows in the shape of an "X." Relationships between major elements enclosed within the heavy box are shown with large double-headed arrows.

The actual relationship of each element within a box to each element of another box is not shown in the figures for the sake of simplicity. When the actual tailoring for the system products is performed for a specific program, these relationships will need to be defined in detail. For example, in Figure 13, the specific relationship between the System/Segment Design Description and the Operational Concept Description in box "Design" to each of the products contained in box "Requirements" is not explicitly shown in the figure. For a specific system with a specific set of specifications, the actual relationships will need to be defined and documented explicitly. This is especially the case where the information that actually represents the products shown in the figure is contained in multiple online sources rather than physical documents.

2. Plan Relationships

The relationships between management plans are described in this section. It is assumed that the "Acquisition Organization" develops an acquisition plan that will be used as a management agreement between the "Acquisition Organization" and its management and between the "Acquisition Organization" and the "User Organization" for the acquisition of the system to be developed. This agreement and those documents referenced by the agreement (e.g., the Acquisition Program Baseline), define the basic philosophy and approach for the acquisition of the system and define the fundamental characteristics the system must meet if the development is to proceed. For the purposes of this report, this agreement is assumed to be documented in the Single Acquisition Management Plan (SAMP), and the other documents are assumed to be included in the SAMP by reference. It is assumed that the "Acquisition Organization" develops a master verification plan to ensure that the system meets its requirements prior to acceptance by the government. For the purposes of this report, this master verification[3] plan is documented in the Test and Evaluation Master Plan (TEMP). The text in subsequent paragraphs of this section refers specifically to elements in Figure 1.

The documentation of the requirements, design, and verification information that describe the actual system to be developed is discussed in Section 3. The documentation of the information required for post-delivery operations and maintenance of the system to be developed is discussed in Section 4. The documents in Sections 3 and 4 are developed as a result of the execution of the processes described in the plans discussed in this section. The plans herein describe the processes to be used to develop the system. The usefulness and quality of the final product are directly related to the processes used by the contractor and the government and the execution of those processes.

The planning products for the system are shown in three boxes in Figure 1. Within the "Formal Agreement Plan" box is the Integrated Master Plan (IMP). This plan, including its associated narrative explanation, is developed by the contractor. In the case of the prime contractor, it is included as part of the development contract when the contract is awarded. This plan contains program characteristics that govern all other planning for the system.

Within the "System Plans" box, plans and schedules are shown as aggregates under the primary topics that must be covered for system acquisition. These aggregate plans and schedules encompass the entire system life cycle and form the basis for lower-level software and hardware development and transition[4] planning and scheduling.

Within the "Development and Transition Plans" box, the software development and software transition plans and schedules encompass the entire computer resources[5] life cycle from system requirements definition through turnover to operations and maintenance and include planning and scheduling for the use of computer resources during software, hardware and system verification, and transition.

The hardware development and hardware transition plans encompass the entire hardware life cycle and include planning and scheduling for the use of computer and non-computer resources hardware during

---

3 The term "verification" rather than "test" has been used consistently throughout to emphasize the fact that the system and its components are verified rather than tested. The term "test" is used only in the context of one of the four verification methods, namely, Inspection, Analysis, Demonstration, and Test.

4 Transition planning is the detailed planning to transfer a product from one organization to another.

5 The term "computer resources" rather than "software" has been used consistently throughout this report to emphasize the fact that software never stands alone outside the context of the hardware system on which it operates. This is true even when the hardware is a commercial product. For software transition planning, it is assumed that the transition of the hardware portion of the computer resources is planned as part of the hardware transition planning and that hardware installation is a prerequisite for software installation.

software, hardware and system verification, and transition. When a specific hardware development does not warrant the creation of a separate hardware development or transition plan, the planning and scheduling for that hardware development or transition is included in or referenced from the planning and scheduling at the system level.

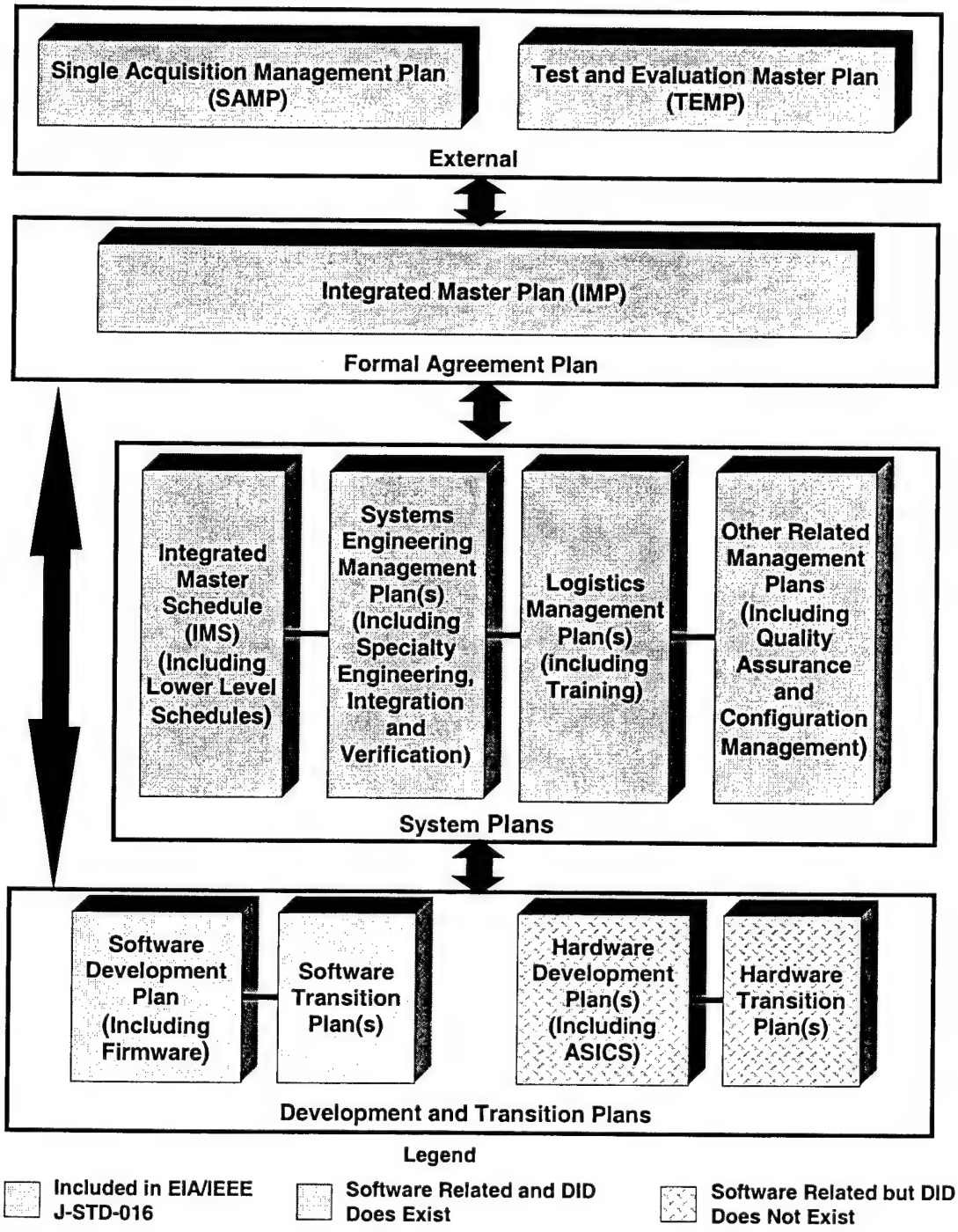


Figure 1. Plan relationships.

2.1 External Plans

2.1.1 Single Acquisition Management Plan

The SAMP defines the acquirer's approach for the acquisition and support of the system. It constitutes an agreement between the acquirer's management, the Acquirer, and the user with respect to what kind of a contracting strategy to use, how competition will be performed and evaluated, and how and when progress will be reported to the acquirer's management and the users.

2.1.2 Test and Evaluation Master Plan

The TEMP defines the acquirer's perspective as to how it will determine that the system is ready for acceptance. This plan constitutes an agreement among the acquirer's management, the acquirer, and the user with respect to the approach for system verification. It also provides guidance to the development organization as to how best to structure its own verification approach. The TEMP focuses on the overall structure, major elements, and objectives of the test and evaluation program that are consistent with the acquisition strategy described in the SAMP. It is intended to provide a road map for integrated simulation, test and evaluation plans, schedules, and resource requirements necessary to accomplish the test and evaluation program. The planning should ensure the timely availability of both existing and planned resources required to support the test and evaluation program.

2.2 System Planning and Scheduling Documentation Introduction

Each acquisition requires an acquisition-specific hierarchy of planning and scheduling products. For system development, this hierarchy of planning and scheduling products is decomposed into disjoint product groupings that make sense for the specific acquisition. These product groupings generally are structured around the organizations that will be responsible for the different parts of the system (e.g., prime contractor and subcontractor, or program office and manufacturing organization within the same company) and then within a given organization around the IPTs or functional disciplines that need to be applied (e.g., systems engineering, software engineering, logistics). Taken as a whole, these organizational groups[6] cover the development of all portions of the system being acquired. Each organizational group may be structured differently depending on the needs of a particular acquisition. These organizational groups are responsible for the planning and scheduling for that portion of the system assigned to them and for ensuring that their portion of the system is managed consistently with all of the other related plans and schedules (both peer-to-peer and prime-to-subcontractor).

Planning covers the actual implementation of the system, the integration of the system into an operational entity, the verification that the system or portion of the system meets its requirements, and the transition of the system or portion of the system to another responsible organization (e.g., subcontractor to prime contractor or prime contractor to government). The planning for system development, including system integration and transition, will be described in the sections that follow. The verification that the system or portion of the system being acquired meets its requirements is performed either independently of the organization responsible for its development or is performed under rigorous conditions witnessed by independent parties. Verification documents and their relationship to the planning and scheduling documents included in the following sections will be discussed beginning in Subsection 3.6.

Planning documents for each organizational group define the processes and procedures that are to be applied to the disciplines required to develop (i.e., implement, integrate, deliver, and maintain) the system

---

[6] For the purpose of this report, "organizational groups" will be the term used to refer to those entities responsible for the some portion of the system being acquired. For any specific acquisition, the actual structure of organizations, IPTs or disciplines responsible for a portion of the system being acquired will need to be developed.

being acquired. For example, systems engineering plans define the systems engineering processes and procedures that will be used during system development, while quality assurance plans define the quality assurance processes and procedures that will be used. Scheduling documents provide a decomposition of the development effort into a time-ordered network of events and their dependencies. The successful execution of the processes and procedures described in these planning documents on the schedules defined in the scheduling documents result in the development of the system being acquired. Successful execution of the processes and procedures depends on having the proper set of personnel and support infrastructure available at the proper time and in the proper quantity.

The processes and procedure portions of these planning documents constitute the "qualitative" planning information needed to execute the system development. The personnel estimates, staffing profile development, assignment of personnel to properly ordered tasks of appropriate duration, and the preparation of an appropriate infrastructure of hardware and software for the personnel to use constitute the "quantitative" planning information needed to execute the system development. Both are required if the development is to be successful.

The execution of the processes and procedures in these planning documents by the appropriately assigned personnel also results in the development of the specification, design, and verification documentation described in Section 3 and the operations and maintenance documentation described in Section 4.

In order to describe the content and relationships among the planning and scheduling products, a representative system, its associated responsible organizations, and its integration structure have been defined in Subsection 2.3. This system is decomposed into a set of representative planning and scheduling groups that demonstrate how the groups and products interact. The text and figures in this section will be referenced as necessary to support the individual product discussion below. In order for these products to be used for a specific acquisition, the relationships and content described here must be translated appropriately into that acquisition's planning and scheduling groups.

2.3 Representative Planning and Scheduling Documentation Tree

This subsection illustrates the relationships among the various planning and scheduling products in the context of the representative system and organizations as shown in Figures 2 through 10. This representative system provides the framework that can be used to apply the appropriate relationships to a specific system. It assumes that each organizational group will use the traditional set of functional disciplines in its development effort, i.e., systems engineering, logistics, quality assurance, configuration management, software development, and hardware development.

The representative system is a satellite system made up of two segments. One segment is a constellation of several satellites. The other segment is a number of existing ground stations (fixed and mobile) into which the new ground system equipment must be placed. This system will be built by a prime contractor, responsible for the entire system, and two subcontractors.

The prime contractor is organized into a program office that interfaces directly with the government acquisition organization and has the responsibility to execute the contract. This program office also is responsible for a portion of the spacecraft (e.g., communications) and a portion of the ground stations (e.g., communications). A commercial satellite organization in the prime contractor's company, independent of the program office, is responsible for providing the spacecraft bus to which the payload and other program-specific equipment will be mated. Another program office in the prime contractor's company, responsible for a different contract, is to provide command and control workstations to be reused on this contract.

The first subcontractor is responsible for the satellite payload and the associated payload ground processing subsystem. This subcontractor is organized into two independent sub-organizations. The first

sub-organization acts as a subcontractor program office and interfaces directly with the prime contractor in the same way the prime contractor interfaces with the government. This sub-organization also is responsible for the development of the payload itself. The second sub-organization, independent of the first sub-organization, is responsible for the payload ground processing.

The second subcontractor acts as a subcontractor program office and interfaces with the prime contractor in the same way the prime contractor interfaces with the government. This subcontractor is responsible for the computer resources infrastructure within the ground stations that allows the ground station equipment to communicate. See Figure 2 for a pictorial representation of these relationships.

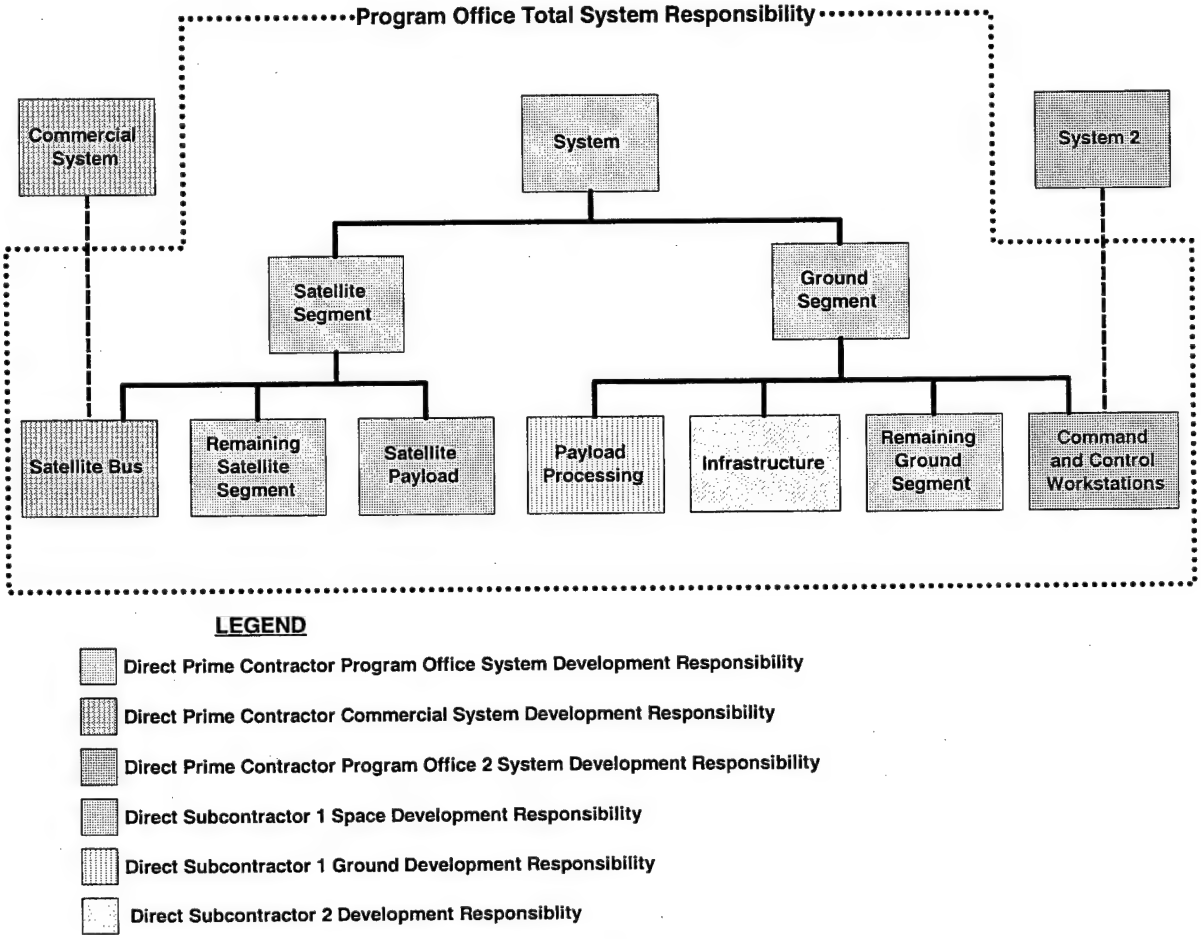


Figure 2. Representative system responsibilities.

2.3.1 Formal Agreement Relationships

Due to the independence of organizations participating in the development of the representative system, this system will require a set of formal agreements between the parties. The set of formal agreements needed for the representative system is as shown in Figure 3. These formal agreements are beyond the scope of this report and will be discussed only in the context of how they impact the planning and scheduling documentation tree. The formal agreements are the mechanisms whereby binding commitments are entered into between parties. These commitments include funding levels and the scope of responsibility and authority relative to their portion of the system being developed. For the purposes of this report, it is assumed that these binding agreements contain the same type of information contained in

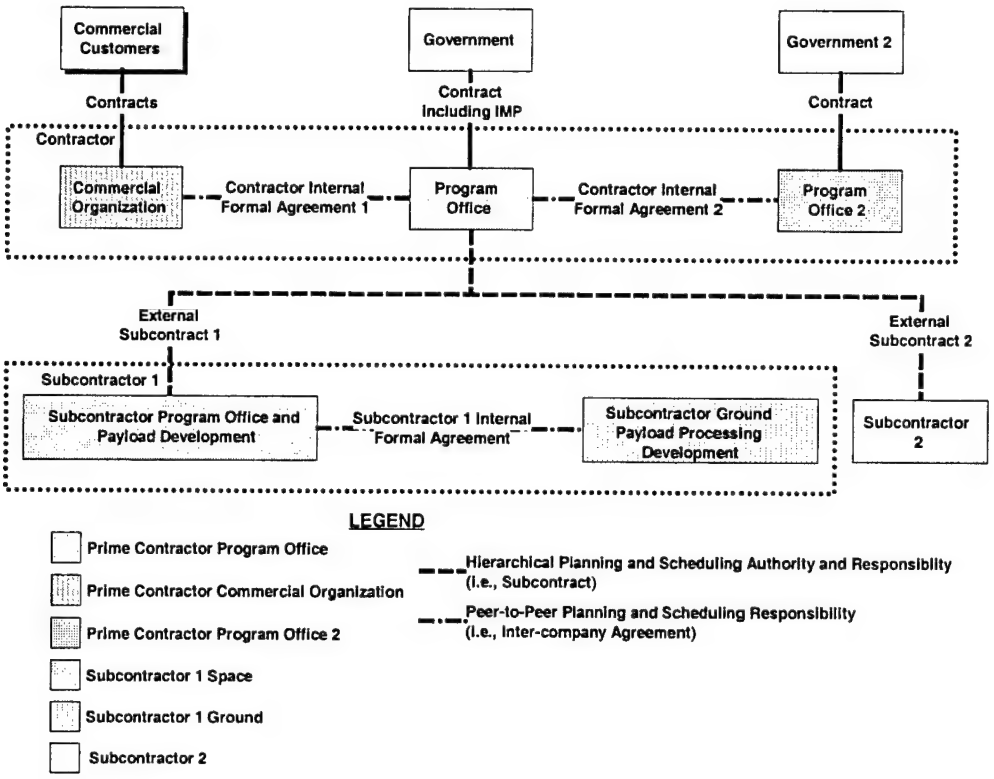


Figure 3. Formal agreement relationships.

the Integrated Master Plan.[7] See Subsection 2.4 for more detailed information. Included in these agreements are tasks to be performed; products to be delivered; processes, procedures, and standards to be followed; lists of events that must occur; lists of accomplishments that must be performed to satisfy each event; and accomplishment criteria used to judge whether an accomplishment has been completed successfully. In addition, it is assumed that schedules and their network of dependencies also are established in the same manner as the Integrated Master Schedule, including the rules for how schedule changes will be handled.

These formal agreements define spheres of responsibility for the planning and scheduling of the system development (including system integration, verification, and transition) and define how the organizations will work together. The spheres of responsibility for the elements of the representative system are shown in Figure 4. Commitments to perform specific management and technical tasks under specific conditions, using specific processes and resources, are documented in the formal agreements. Once initiated, these formal agreements define the characteristics that govern all other management and technical interactions needed to develop the system.

Informal interactions are the normal day-to-day management and technical interactions required to develop the system. These day-to-day management and technical interactions are performed consistently with the structure defined in the formal agreements. When one or more of the parties to a formal agreement perceive that some informal interaction or some condition within their sphere of responsibility conflicts with the formal agreement, then new formal interactions must take place to resolve the issue.

---

[7] When these binding agreements take on different forms and content, there is a greater likelihood that elements critical to the successful delivery of the system will be omitted. For example, tasking to perform software safety criticality analysis might be overlooked in the subcontractor tasking. Disconnects such as these will severely lower the probability of successful project completion.

This may result in one or more changes to the formal agreement that then impact other aspects of the system (e.g., technical responsibilities or schedules).

For the purpose of this report, it is assumed that these formal agreements are in place, that the appropriate set of tasks, processes, procedures, standards, and so forth have been included, and these agreements will not need to be modified.[8]

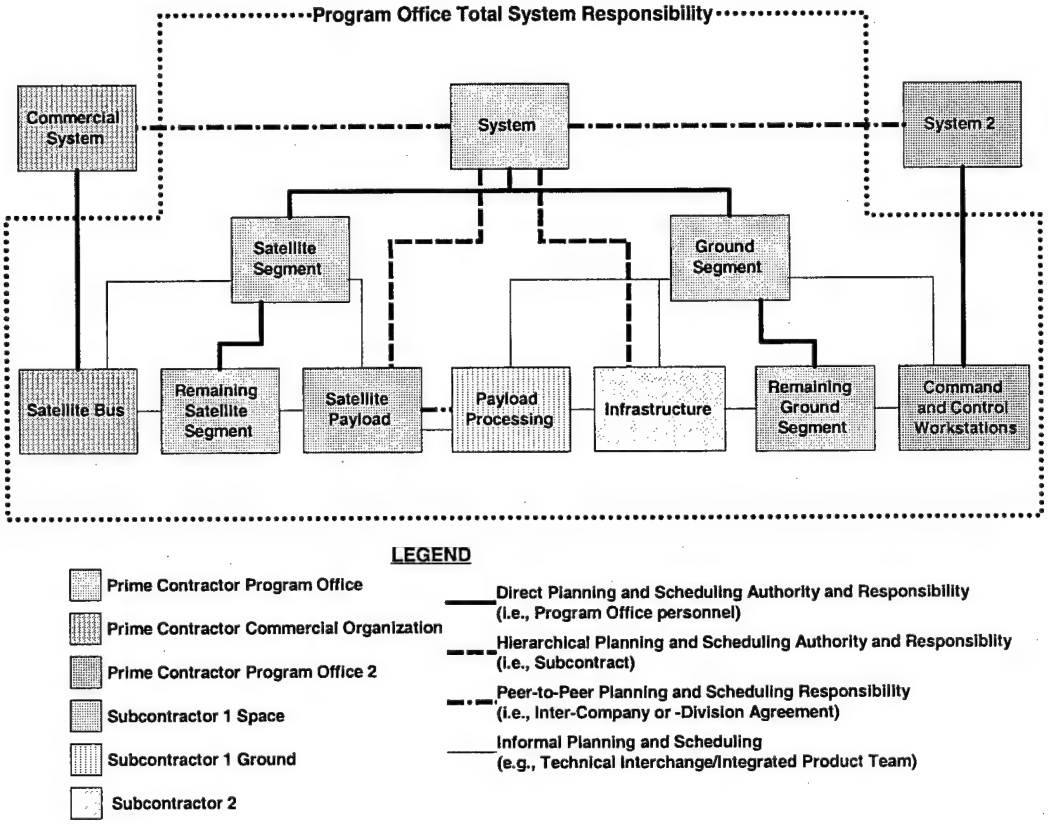


Figure 4. Representative system formal and informal interactions.

2.3.2 Interactions Among Organizations and Planning Documents

Each organizational group that requires a formal agreement also requires a set of planning and scheduling documents. These documents define the processes, procedures, and schedules applicable to that portion of the system development under the control of that organizational group. As shown in Figure 3, this representative system has both peer-to-peer (e.g., program office and the commercial organization) and hierarchical relationships (e.g., program office and Subcontractor 1) for the planning and scheduling documents. Whenever an organizational group has peer-to-peer or hierarchical relationships with other organizational groups as well as direct responsibility for the development of a portion of the system, the planning and scheduling document set must encompass all of these relationships either as sections within a single document or as multiple documents. For example, as shown in Figure 3, the program office organization has peer-to-peer relationships with the commercial organization and Program Office 2. In

[8] In reality, significant risks are introduced into the system development by not having the appropriate set of tasks, processes, procedures, standards, and so forth correctly included in these formal agreements. For example, a subcontract may inadvertently omit tasking for software safety or reliability analysis, or the tasking may not require participation in the prime contractor's Integrated Product Teams (IPTs). Considerable effort should be expended to ensure that the formal agreements contain all pertinent information when they are executed.

addition, it has a hierarchical relationship to both Subcontractors 1 and 2. As shown in Figure 4, the Program Office has direct development responsibility for portions of the Ground Segment and the Satellite Segment. As a result of these relationships, the set of planning and scheduling documents must account for the specific processes, procedures, and schedules associated with each peer-to-peer relationship, each subcontractor relationship, and the specifics of their own development responsibility.

See Figure 5 for an illustration of the set of planning and scheduling documents that correspond to the formal agreement structure shown in Figure 3. For the sake of simplicity, this figure shows only the relationships for the formal agreements and a consolidated set of system plans. The same set of relationships is applicable to the hardware and software planning as well. In all cases, these planning and scheduling documents must be compatible with each other and must be kept consistent with each other as development progresses.

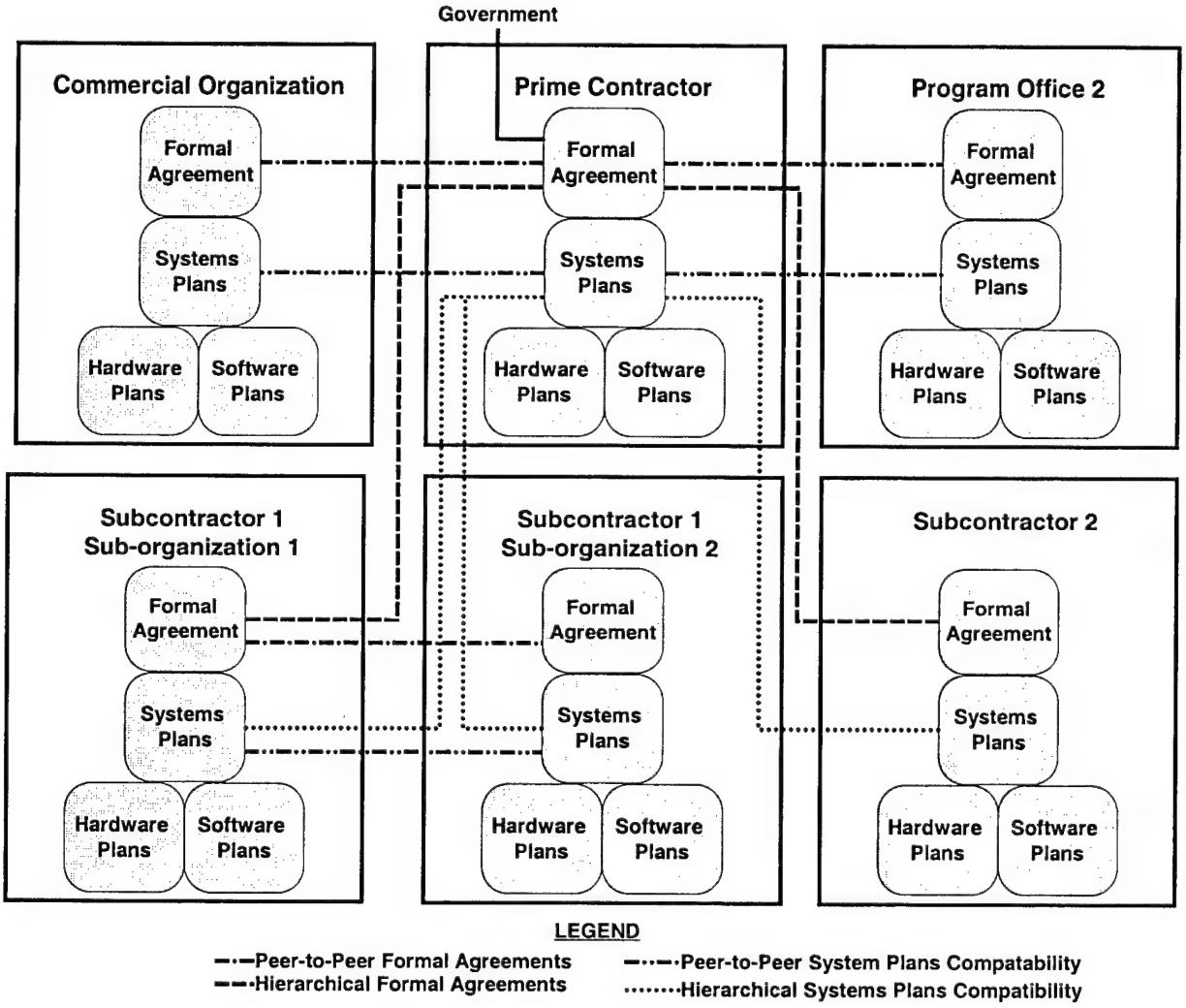


Figure 5. Organization and planning document interaction.

Since the set of planning and scheduling documents is the same for each organizational group, this report will describe the document set as if there were only one set of these documents. In addition, since our assumption is that the formal agreements have the same content as the IMP and IMS, this report will describe all formal agreements together in the same section as if each contained IMPs and IMSs. See Figure 6 for the single set of planning and scheduling documents to be described. This figure shows the

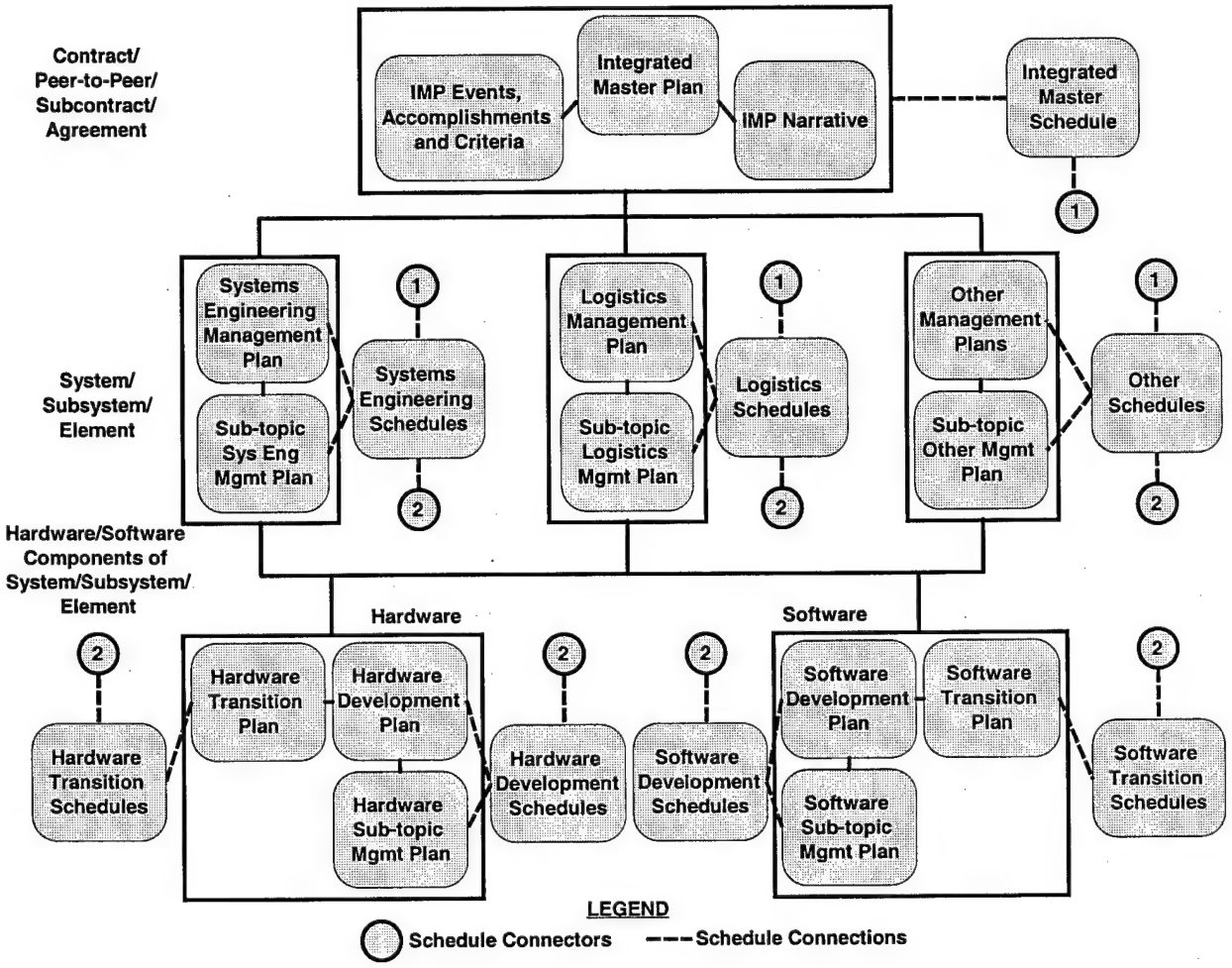


Figure 6. Planning and scheduling hierarchy for each formal agreement.

relationship between the formal agreement and the system plans and between the system plans and the hardware and software plans. The plans and schedules in the figure are represented as a hierarchy. For clarity of presentation, the planning documents are connected directly to each other, while the schedules are connected to each other via connectors. To apply these descriptions to a specific program, the following would need to be determined: the organizational groups defined by the set of formal agreements, the type of relationship between the organizational groups (i.e., peer-to-peer versus hierarchical), and the set of planning and scheduling documents required for each organizational group (e.g., is software planning required or is only hardware planning needed?). Finally the set of planning and scheduling documents would need to be developed that explicitly accounted for each of the relationships for each of the planning and scheduling primary topic aggregates and sub-topics shown in Figure 1 (e.g., System Engineering, Logistics, Hardware, and Software).

Although each organizational group has essentially the same hierarchy of planning and scheduling documents, the scope of required planning and scheduling will differ depending on the performing organization. For example, the planning and scheduling done by the prime contractor must encompass the entire system, including all peer organizations and subcontractors, as well as its own direct development responsibilities, while the planning and scheduling performed by Subcontractor 2 encompasses only its portion of the system. However, the Subcontractor 2 planning and scheduling must be consistent with the prime contractor's planning and scheduling.

Planning and scheduling by each organizational group consists of similar responsibilities. For systems engineering, and hardware and software development and transition, "qualitative" plans (e.g., processes, procedures, and standards) and "quantitative" plans (e.g., effort estimates, staffing profiles, task identification and scheduling, personnel assignments to tasks) are developed and documented. Their execution results in:

a. *Development of a hardware and software architecture that meets the requirements of the system or portion of the system that is the responsibility of that organizational group.* The system hardware and software architecture is documented in system design documents described in Subsection 3.5.1.

b. *Decomposition of the architecture into individual components (hardware and software) from which the system will be constructed.* These architectural components are described by a hierarchical set of documents that define the requirements that individual components must meet and a set of design documents, derived from the requirements documents, from which the actual components will be built. The hierarchical set of requirements documents, called a specification tree, is described in Subsection 3.4. The set of design documents is described in Subsection 3.5.2.

c. *Development/procurement/reuse and scheduling of the actual individual hardware and software architectural components.* This includes the performance of design verification using documented design verification plans, descriptions, and reports.[9] Design verification ensures that the architectural components accurately reflect their designs and requirements prior to integration with other architectural components. These design verification plans, descriptions, and reports generally are documented in engineering documents (e.g., software development folders/files or hardware engineering notebooks) at the lowest level of the system development hierarchy. These documents should be maintained throughout the life of the system and updated as design changes are made. These documents are beyond the scope of this report and will not be discussed further.

d. *Determination of what order and when to integrate the capabilities of individual architectural hardware and software components into compound components to construct the system or portion of the system.* The order of integration for the system, or portion of the system and the integration schedule, are determined by many factors, including the system architecture, the organizational groups responsible for the architectural components, and the choice of life cycle model to be used for the architectural components' development (e.g., waterfall, evolutionary, spiral). Each integration effort in this order of integration is called an integration level. These integration levels may or may not be a strict hierarchical structure of architectural components, depending on the needs of the individual program. However, these integration levels do represent an increasing level of functionality and complexity of hardware and software components as more and more functions and components are combined. Each integration level is used to determine that the set of capabilities planned for that level performs as designed and meets its requirements. Taken as a whole across all organizational groups, the set of all integration levels is structured to ensure that all system capabilities have been implemented successfully and all requirements have been met. The actual integration performed at each integration level is based on documented integration plans and descriptions, and the results of the integration are documented in reports. In general, the plans, descriptions, and reports are developed by the organization responsible for performing the integration and documented in engineering documents (e.g., system or software integration folders/files or

[9] Plans, descriptions, and reports are general terms used to describe the set of information needed to perform any verification process. Plans state the verification goals (e.g., what the verification will accomplish using what resources). Descriptions define, first, the set of actual verification cases for each verification method planned to accomplish the verification goals (e.g., the set of test and analysis cases needed to determine goal accomplishment) and second, the actual steps required to perform individual verification cases (e.g., the set of test procedure steps in each test case) in sufficient detail to allow repeatability. Reports document the results of the verification. See Subsection 3.6 for additional information on plans, descriptions, and reports.

engineering notebooks, and integration problem reports or integration summaries). Integration planning for the system as a whole is described in Subsection 2.5.2. Integration planning for hardware and software is described beginning in Subsection 2.6.

e. *Determination of which integration levels to use to demonstrate formally to independent observers that the requirements allocated to the hardware and software functions and components contained in that integration level have been met.* In general, only a subset of the integration levels is used to demonstrate formally that requirements are met. This subset is determined by many factors, including the system architecture, the organizational groups responsible for the architectural components, and the choice of life cycle model to be used for the architectural components' development (e.g., waterfall, evolutionary, spiral). Each verification effort associated with an integration level is called a verification level. Taken as a whole across all organizational groups, the set of all verification levels is structured to ensure that independent observers have determined that all system requirements have been met successfully. The actual formal verifications performed at each verification level are based on documented verification plans and descriptions, and the results are documented in reports. In general, the verification plans, descriptions, and reports are developed and documented by the organization responsible for performing the formal verification. The relationship of verification levels to requirements specifications, as well as information on the formal verification plans, descriptions, and reports are described beginning in Subsection 3.6.

f. *Determination of what order to transition (i.e., transfer) hardware and software architectural components from one responsible organization to another (including between contractors and between the prime contractor and the acquirer's user and maintenance organizations).* The order of transition of hardware and software architectural components is dependent on many factors, including whether the transfer is an interim capability or the final product, the maintenance philosophy contained in the formal agreement between organizations, and the choice of life cycle model to be used for the architectural components' development (e.g., waterfall, evolutionary, spiral). In general, hardware and software architectural components are transitioned from the implementing organizational group to an integrating organizational group until the final transition of the system to the acquiring organization (i.e., to the appropriate operations and maintenance organizations). The transition occurs following the completion of a subset of the planned integration levels or combined integration and verification levels. Taken as a whole across all organizational groups, the set of transitions is structured to ensure that, following final system acceptance, all contracted hardware and software system capabilities are transferred to the acquiring organization. The actual transitions performed following the selected integration or combined integration and verification levels are based on documented hardware and software transition plans. Problems encountered during the transition are documented in problem reports. In general, the hardware and software transition plans and problem reports are developed and documented by the organization responsible for performing the transition. Transition planning for the system as a whole is described in Subsection 2.5.2. Transition planning for hardware and software is described beginning in Subsection 2.6.

For the logistics area, "qualitative" plans (e.g., processes, procedures and standards) and "quantitative" plans (e.g., effort estimates, staffing profiles, task identification and scheduling, personnel assignments to tasks) are developed and documented; their execution results in:

a. *Determination of the supportability objectives and supportability-related design requirements for the new system; how best to implement the supportability requirements; the logistic support resource needs; and that the system is, in fact, supportable at startup and throughout the life cycle.* Supportability planning must encompass both hardware and software components. Supportability planning focuses on how logistics personnel will work with development personnel to optimize the support system to achieve the best balance between cost, schedule, performance, and supportability. Supportability planning is described in Subsection 2.5.3.1.

b. *Determination of what operations and maintenance training courses are needed; the order to perform the training and the personnel requiring the training; and the coordination of the training needs (e.g., training system components) with system development.* The planning for the operations and maintenance training needs for the system as a whole is described in Subsection 2.5.3.2. The planning for the training needs of development personnel is included as part of the management plans appropriate to the development personnel needing training (e.g., SEMP, Software Development Plan (SDP) and logistics).

For the other areas, "qualitative" plans (e.g., processes, procedures and standards) and "quantitative" plans (e.g., effort estimates, staffing profiles, task identification and scheduling, personnel assignments to tasks) are developed and documented. Their execution results in:

a. *Determination of what constitutes quality for system products, how quality will be monitored and reported effectively during product development, and how corrective action will be monitored when nonconformance is identified.* The quality assurance planning is described in Subsection 2.5.4.1.

b. *Determination of what system components and documentation should be controlled; how to identify them; how to control them; how to report on their status; and how and when to perform configuration audits.* The configuration management planning is described in Subsection 2.5.4.2.

2.3.3 Integration, Verification, and Transition Relationships

For the purpose of this report, suppose that in the representative system, systems engineering planning and formal agreements have established the need for the following development and integration facilities:

a. The Prime Contractor Program Office has a satellite integration facility that will be used for integration and verification of the satellite segment hardware and software. This facility also will be used for the development, integration, and verification of the satellite hardware and software under its direct development responsibility. The Program Office also has a ground segment integration facility that will be used for integration and verification of the ground segment hardware and software. This facility also will be used for the implementation, integration, and verification of the ground segment hardware and software under its direct development responsibility. The Prime Contractor Program Office satellite and ground segment integration facilities are connected to allow full system integration and verification prior to shipment to the launch site and ground stations.

b. The Prime Contractor Commercial Organization has a satellite integration facility for the integration and verification of the satellite bus hardware and software prior to delivery to the Prime Contractor Program Office's satellite integration facility.

c. The Prime Contractor Program Office 2 has a ground integration facility where the integration and verification of its system will be performed prior to shipment to its customer and that will be used for determining that its command and control workstations are ready for transfer to the Prime Contractor Program Office's ground segment integration facility.

d. Subcontractor 1's Payload Development Organization has a general-purpose payload integration facility to be used for the implementation, integration, and verification of the payload hardware and software under its direct responsibility.

e. Subcontractor 1's Ground Payload Processing Organization has a general purpose ground processing integration facility to be used for the implementation, integration, and verification of the payload ground processing hardware and software under its direct responsibility.

f. Subcontractor 2 has a general-purpose development facility that will be used to develop the ground segment infrastructure, but this subcontractor will use the Prime Contractor's ground segment integration facility to perform its integration and verification.

See Figure 7 for a pictorial representation of these relationships. The actual organization and relationship among these facilities must be determined for each specific program.

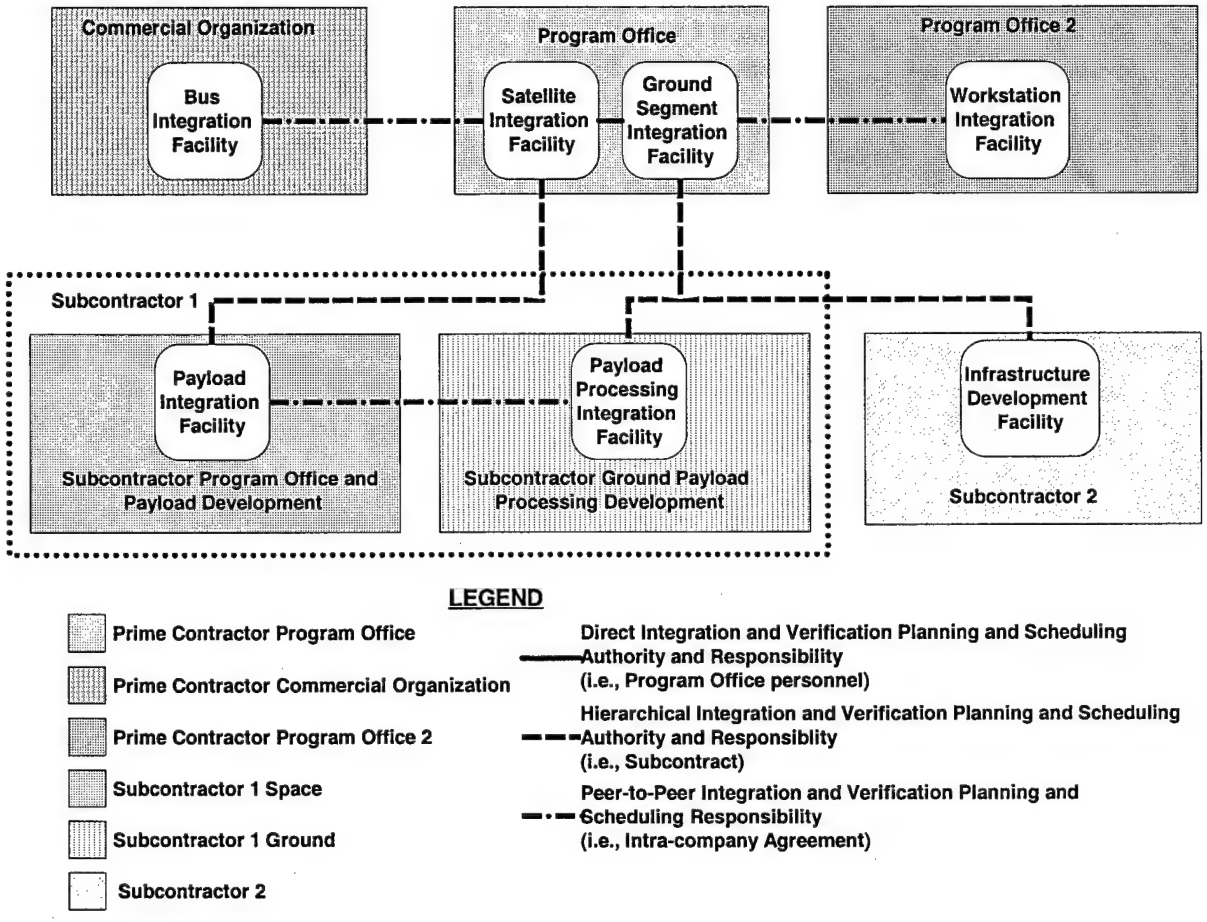


Figure 7. Integration and verification facility relationship.

In the interest of simplicity, only a portion of the integration, verification, and transition planning and scheduling relationships in the representative system will be described in detail. All parties participating in the system development will have similar sets of relationships. The portion chosen for detailed discussion consists of the planning and scheduling relationships between the Program Office and Subcontractor 1. As shown in Figure 2, Subcontractor 1 is responsible for the payload that will be attached to the satellite and the payload ground processing of the data collected.

Performing the planning and scheduling processes and procedures contained in the respective organization's systems engineering plans, hardware and software development plans, and hardware and software transition plans will result in formalized agreements among the various affected parties for a set of incremental products and their associated schedules. In parallel with formalizing the incremental products and schedules, other planning information is developed and added to the various planning and scheduling documents. This other planning information includes the identification of personnel and their assignment to tasks, and the identification of required non-personnel resources (e.g., test equipment or development environments) and planning for their acquisition/development/reuse. Once determined, this

information becomes part of the planning and scheduling information used by each organizational group to manage the development of its portion of the system.

For the purposes of this report, it is assumed that each party has sufficient resources to complete the work as scheduled. In most system developments, changes that impact resources and schedules are occurring constantly, and it is essential that the planning and scheduling processes and procedures are able to effectively handle this constant change.

The following discussion references Figure 8, which depicts the development, integration, verification, and transition relationships for Subcontractor 1. Where appropriate to the discussion, associations will be made between the text and different portions of the figure using corresponding numbers in each.

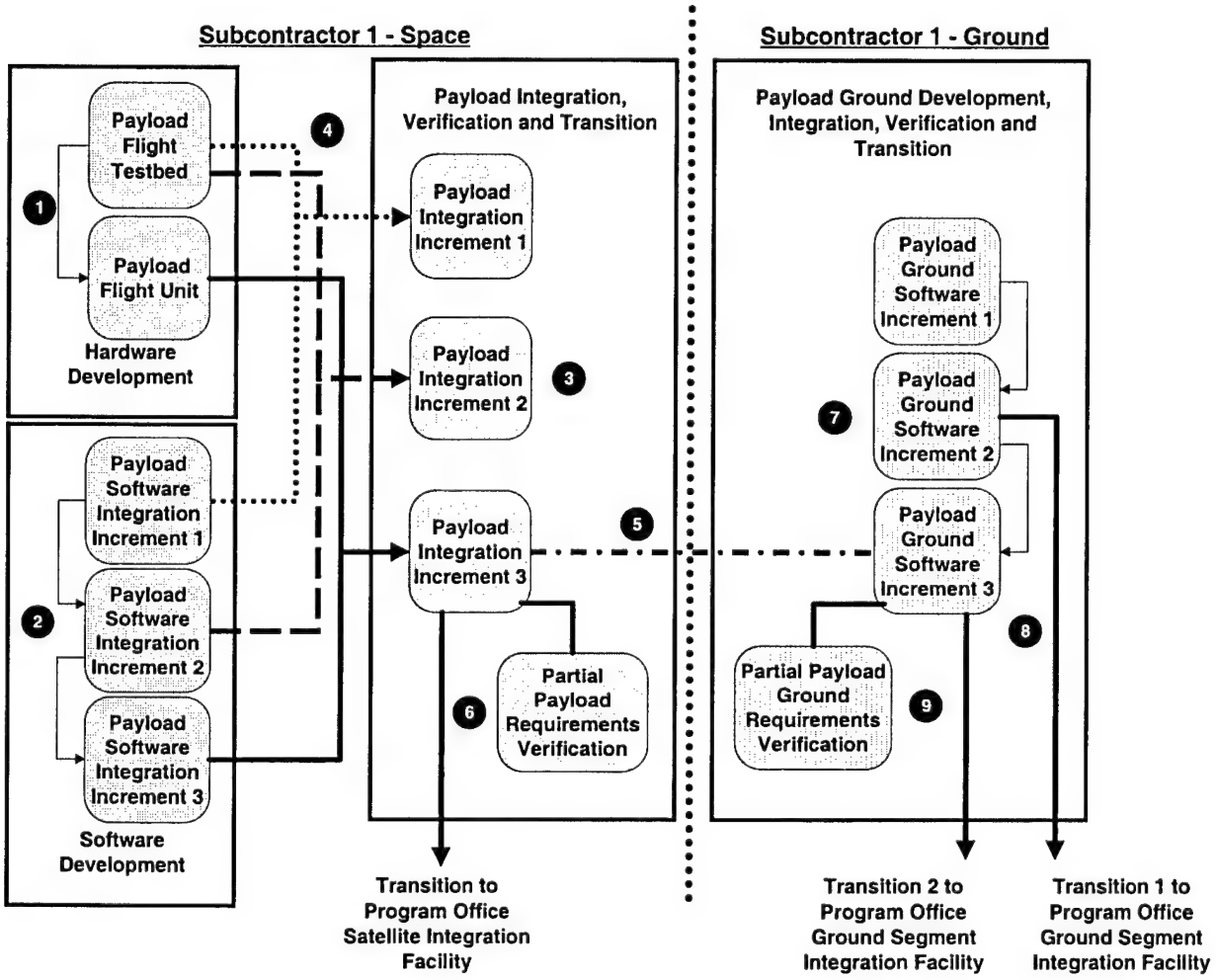


Figure 8. Subcontractor 1 development, integration, verification, and transition relationships.

Formal Agreement: The Program Office has performed its planning and scheduling function and entered into a formal agreement with the space organization of Subcontractor 1 for the development (i.e., implementation, integration, verification, and transition to the Program Office) of the hardware and software components that comprise the satellite payload and its associated payload ground processing. The Subcontractor 1 space organization has performed its planning and scheduling function and entered into a formal agreement with the ground organization for the development of the hardware and software

components that comprise the payload ground processing subsystem. The Subcontractor 1 ground organization has performed its planning and scheduling function and is prepared to perform its development as called for in its formal agreement with the Subcontractor 1 space organization.

Subcontractor 1 - Space: Based on the systems engineering and hardware/software planning, it is determined that the most efficient method for meeting its formal agreements is to develop the hardware in two increments (hardware integration increment levels 1 and 2) ❶ and its software in three increments (software integration increment levels 1, 2, and 3) ❷. The arrows joining hardware and software increments indicate increasing complexity and functionality. The hardware and software development organizations will perform their development based on the processes and procedures contained in their hardware and software development plans and the personnel tasking, and so forth, derived from that planning.

The hardware and software increments will be integrated by systems engineering in three increments (payload system integration increment levels 1, 2, and 3) ❸. Figure 8 shows the contents of each integration increment using different types of lines ❹. The systems engineering/integration organization will perform the hardware and software integration based on the processes and procedures contained in the systems engineering management plan. The results of this planning have integration increment 3 designated (i.e., planned and scheduled) to be used for informal Subcontractor 1 integration testing between the payload and ground processing prior to final hardware and software transition to the Program Offices' integration facilities (hardware and software transition level 1) ❺.

In addition, integration increment 3 will be used for formal verification of a portion of the payload requirements (payload verification level 1), and will be the item (hardware and software) to be transitioned into the Program Office's Satellite Integration Facility ❻.

Subcontractor 1 - Ground: Based on the software planning, it is determined that the most efficient way to meet its formal agreements is to develop the software in three[10] increments (ground software integration increment levels 1, 2, and 3) ❼. As with the space payload software, the arrows connecting the increments indicate increasing complexity. For the ground processing, it is assumed that commercial off-the-shelf (COTS) computer hardware will be used, so no separate hardware development is shown. The software development organization will perform its development and integration based on the processes and procedures contained in its software development plan. The results of this planning have integration increment 3 designated to be used for informal Subcontractor 1 integration testing prior to final transition to the Program Offices' Ground Segment Integration Facility ❺. Integration increment 2 will be transitioned to the Ground Segment Integration Facility for initial integration testing (transition level 1) ❽.

Increment 3 will be used for formal verification of a portion of the payload ground processing requirements (verification level 1) and will be the item to be used for final transition into the Program Office's Ground Segment Integration Facility (transition level 2) ❾.

The transition planning for both the Space and Ground organizations must ensure that sufficient capability still exists at their own facilities following transition to allow continued development of additional increments and efficient maintenance when anomalies occur during Program Office integration. Alternatively, arrangements must have been made for continued development and integration support at the Program Office's Integration Facilities.

---

[10] It is only an artifact of the example that both the payload and ground software here and the subsequent Program Office integration for space and ground have three increments. In general, the number of increments in each is different.

The following discussion references Figure 9, which depicts the development, integration, verification, and transition relationships for Subcontractor 1. Where appropriate to the discussion, associations will be made between the text and different portions of the figure using corresponding numbers in each.

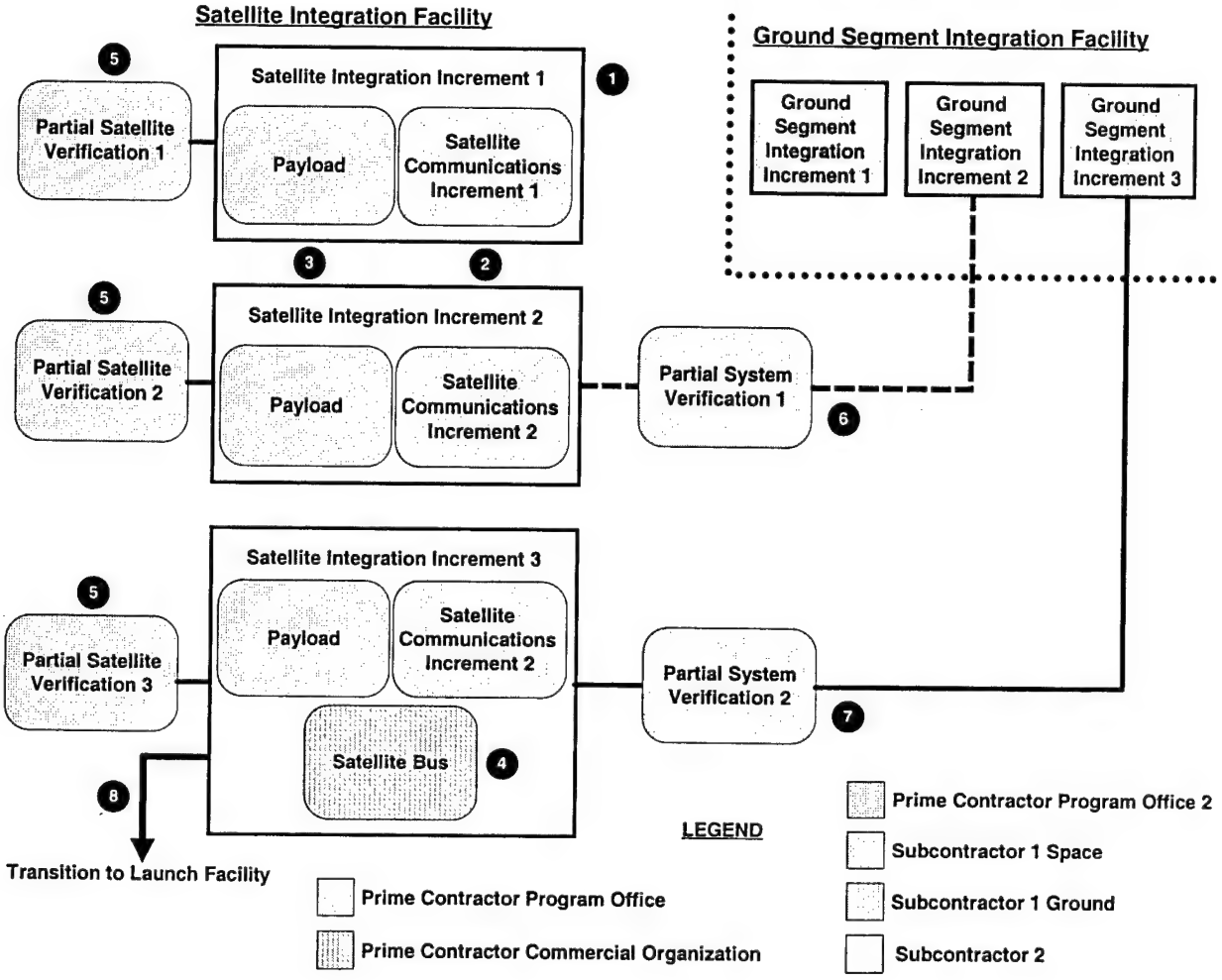


Figure 9. Program office integration, verification, and transition relationships - part 1.

Formal Agreement: The Program Office has performed its planning and scheduling function and entered into formal agreements with the Space Organization of Subcontractor 1 and the Commercial Organization for the development (i.e., implementation, integration, verification, and transition) of the hardware and software components appropriate to each.

Program Office Satellite Integration Facility: Based on systems engineering and hardware/software planning, it is determined that the most efficient method for meeting its formal agreement for the satellite is to integrate the satellite hardware and software in three increments ❶. The hardware and software development organizations in each organizational group contributing to the final satellite will perform their development based on the processes and procedures contained in their hardware and software development plans.

The systems engineering organization in each organizational group will integrate that organizational group's contribution and coordinate the hardware and software transition to the Satellite Integration

Facility according to the processes and procedures contained in their systems engineering management plan. (See Figure 8 for how this is performed for Subcontractor 1 - Space.)

In the representative system, the Program Office development group provides two increments of the satellite communication hardware and software ❷. Subcontractor 1 - Space provides one final version of the payload hardware and software ❸, while the Commercial Organization provides one final version of the satellite bus hardware and software ❹.

In the case of the Program Office, its systems engineering management plan must contain the planning for the integration of the hardware and software it is responsible for developing (e.g., the satellite communications) as well as the planning for the integration of the development from each of the other contributing organizational groups.

Each integration increment is planned to be used for formal verification of a portion of the satellite requirements ❺. In addition, Satellite Integration Increment 2 will be used for early satellite to ground segment integration as well as for formal verification of a portion of the system requirements ❻. Satellite Integration Increment 3 will be used for final integration of the satellite and ground segment ❼ prior to transition of the satellite to the Launch Facility ❽ and the ground segment to the ground station. In addition, Integration Increment 3 will be used for the formal verification of a portion of the system requirements ❼.

The transition planning within each contributing organization must ensure that sufficient capability still exists at their own facilities following transition to allow continued development of additional increments and efficient maintenance when anomalies occur during Program Office integration. Alternatively, arrangements must have been made for continued development and integration support at the Program Office's integration facilities. The transition planning at the Program Office must ensure that sufficient capability exists at their own facility or at the appropriate contributing organization's facilities following transition to allow continued development and efficient maintenance when anomalies occur at the user's site.

The following discussion references Figure 10, which depicts the Program Office integration, verification, and transition relationships, focusing on the ground segment integration levels. Where appropriate to the discussion, associations between the text and different portions of the figure will be made using corresponding numbers in each. Although the details of the planning and scheduling in each organizational group are not provided, they are similar to those for Subcontractor 1.

Formal Agreement: The Program Office has performed its planning and scheduling function and entered into formal agreements with the Space Organization of Subcontractor 1, Subcontractor 2, and Program Office 2 for the development (i.e., implementation, integration, verification, and transition) of the hardware and software components appropriate to each. The Subcontractor 1 Space Organization has performed its planning and scheduling function and entered into a formal agreement with the Subcontractor 1 Ground Organization for the development of the hardware and software components that make up the payload ground processing subsystem.

Program Office Ground Segment Integration Facility: Based on the systems engineering and hardware/software planning, it is determined that the most efficient method for meeting its formal agreement for the ground segment is to integrate the ground segment hardware and software in three increments (ground segment integration levels 1, 2, and 3) ❶.

The hardware and software development organizations in each organizational group contributing to the final ground station will perform their development based on the processes and procedures contained in their hardware and software development plans.

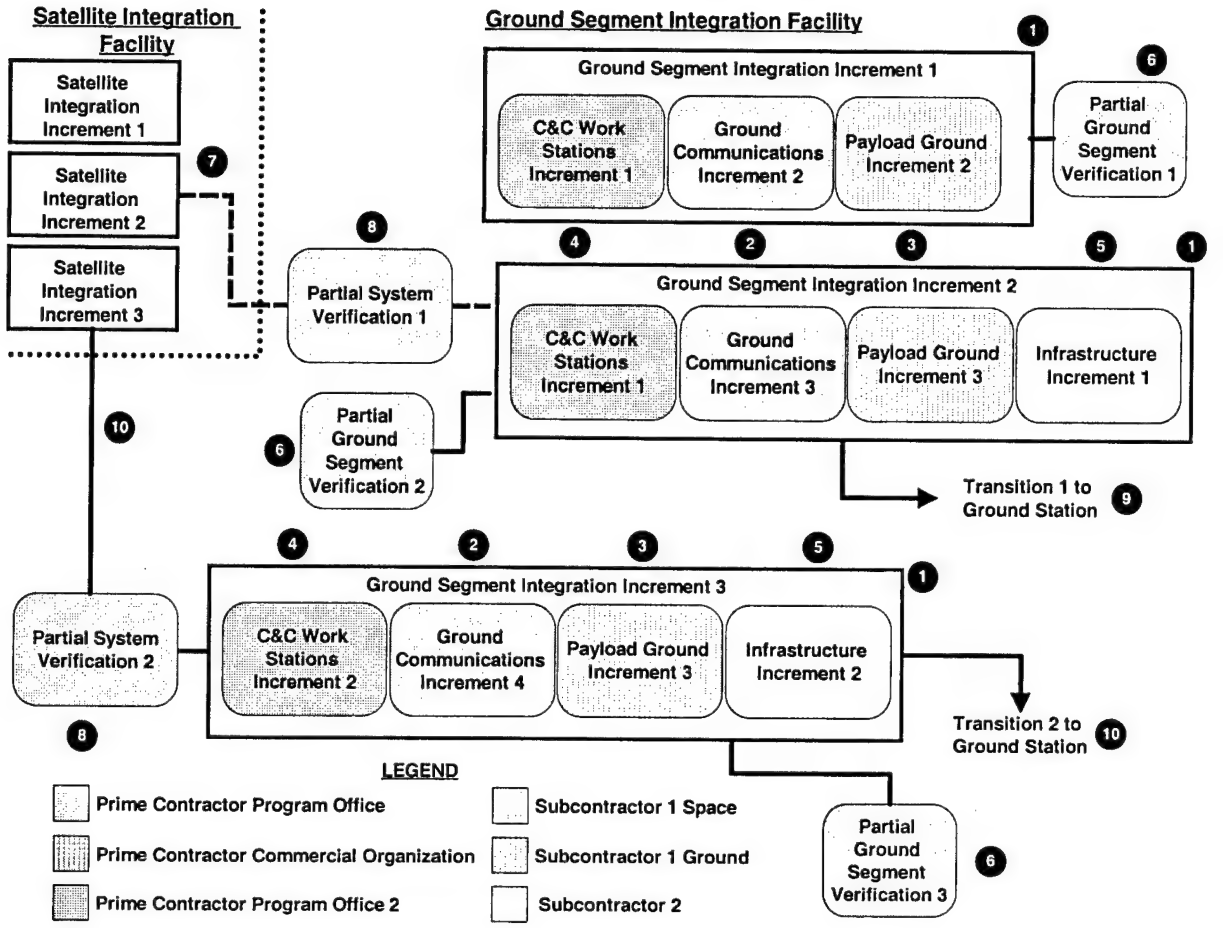


Figure 10. Program office integration, verification, and transition relationships — part 2.

The systems engineering organization in each organizational group will integrate that organizational group's contribution and coordinate the transition of it to the Ground Segment Integration Facility according to the processes and procedures contained in their systems engineering management plans. See Figure 8 for how this is performed for Subcontractor 1 - Ground.

In the representative system, the Program Office development group provides three increments of the ground communication hardware and software (communications integration levels 2, 3 and 4[11]) ❷. Subcontractor 1 - Ground provides two increments of the payload ground processing hardware and software (payload ground processing integration levels 2 and 3) ❸. Program Office 2 provides two increments of the command and control (C&C) workstation hardware and software (C&C workstation integration levels 1 and 2) ❹. In addition, Subcontractor 2 provides two increments of the Infrastructure hardware and software (infrastructure integration levels 1 and 2) ❺.

In the case of the Program Office, its systems engineering management plan must contain the planning for the integration of the hardware and software it is responsible for developing (e.g., the ground communications) as well as the planning for the integration of the development from each of the other contributing organizational groups.

[11] Components with increment numbers smaller than those shown in the Ground Software Integration Increments (e.g., Ground Communications Increment 2 in Ground Segment Integration Increment 1) were performed internal to that specific development organization and occurred prior to the Ground Software Integration Increment of which they are a part.

Each integration increment is planned to be used for formal verification of a portion of the ground segment requirements (ground segment verification levels 1, 2, and 3) ❻. In addition, Ground Segment Integration Increment 2 will be used to perform early satellite-to-ground segment integration (system integration level 1) ❼, the formal verification of a portion of the system requirements (system verification level 1) ❽, and will be transitioned to the user's ground station for initial integration there (ground segment hardware and software transition 1) ❾. Ground Segment Integration Increment 3 will be used for final integration of the satellite and ground segment ❿ (system integration level 2) prior to transition of the satellite to the Launch Facility and the ground segment to the ground station ❿. In addition, Ground Segment Integration Increment 3 will be used for the formal verification of a portion of the system requirements (system verification level 2) ❽.

Note that for both the Ground Communications and Payload Ground Processing, the initial integration was performed within the context of the development organizational group prior to transition to the Ground Segment Integration Facility. See Figure 8 for how this is performed for Subcontractor 1.

The transition planning within each contributing organization must ensure that sufficient capability still exists at their own facilities following transition to allow continued development of increments and efficient maintenance when anomalies occur during Program Office integration. Alternatively, arrangements must have been made for integration support at the Program Office's integration facilities. The transition planning at the Program Office must ensure that sufficient capability exists at their own facility or at the appropriate contributing organization's facilities following transition to allow continued development of increments and to allow efficient maintenance when anomalies occur at the user's site.

2.4 Formal Agreement Plan: Integrated Master Plan

The IMP is an event-driven plan that documents the significant accomplishments necessary to develop that portion of the system assigned to a particular organizational group. For the prime contractor, the IMP is developed from the "Statement of Objectives" (SOO) or the "Statement of Work" (SOW) and includes the entire system. The SOO and SOW are developed by the government and documented in the "Request for Proposal" (RFP), or "Call for Improvement" (CFI). The IMP is the primary tool used to track program status and is prepared by the organization responsible for the development. For the prime contractor, the IMP covers the entire program, including all peer organizations and subcontractors.

The IMP consists of two main sections: the IMP Narrative and the IMP events, significant accomplishments, and accomplishment criteria. The IMP Narrative describes the high-level processes to be used for the major disciplines (e.g., systems engineering, software development, specialty engineering, logistics) and specifies the compliance documents that will be used for system development. For each significant system event (e.g., system design reviews, mission ground station integration complete), the IMP Event Table identifies the significant accomplishments that must be completed for that event and defines the criteria each accomplishment must meet in order to be completed successfully. All lower-level planning and scheduling, including hardware and software, must be consistent with the information contained in this plan. It is critical that the information in the IMP includes all of the meaningful system events, that each event has accomplishments that characterize the completion of that event, and that each accomplishment has meaningful, objective, and verifiable criteria defined. It also is critical that meaningful standards for all of the disciplines needed for system development are included in the IMP (e.g., systems engineering, software development, logistics, training, configuration management) and that these standards are consistent and compatible across all organizational groups responsible for portions of the system to be delivered.

2.5 System Plans

System plans are shown as aggregates under the primary topics that must be covered by these plans. The primary topics are the Integrated Master Schedule, Systems Engineering Management Plan, Logistics Management Plans, and "Other" Management Plans (see Figures 1 and 6). Individual programs or specific organizational groups supporting a specific program may have different groupings of plans. The most critical aspect of these plans is that all relevant topics and sub-topics are planned somewhere and that all of these plans are consistent and compatible, with the IMP, among themselves, and with the appropriate plans in all other related organizational groups.

The primary topic aggregates within each organization's sphere of responsibility may themselves consist of hierarchical levels of planning and scheduling documents depending on the needs of the individual program and how that organization is structured. For example, the representative system is a satellite system consisting of multiple fixed and mobile ground stations and a constellation of satellites (see Subsection 2.3). In this system, the prime contractor may have a systems engineering plan for the system as a whole that covers all topics and sub-topics applicable system-wide, whether the work is being performed in-house or by peer-to-peer organizations or subcontractors. The prime contractor may have separate systems engineering planning for the ground segment and the satellite as a whole that covers all topics and sub-topics applicable segment-wide, whether the work is being performed in-house or by peer-to-peer organizations or subcontractors. The prime contractor also may have two additional systems engineering plans that cover those portions of the ground segment and space segment that are the prime contractor's direct development responsibility. In addition, each primary topic may have different groupings. For example, while there may be separate systems engineering planning for those portions of the ground segment and satellite that are the prime contractor's responsibility, there may be only one quality assurance plan for both.

How the primary topic aggregates are organized is specific to each individual program. For the purposes of this report, we shall assume that each organizational group will have a single plan for each primary topic aggregate and that each sub-topic will be contained either directly in the primary topic plan as a separate section/appendix or in a separate document referenced from the primary topic plan. See Figure 6, which illustrates the assumed structure for the system plans in this report. This figure shows the relationship between the formal agreement and the system plans and between the system plans and the hardware and software plans. Each primary topic and sub-topic in the system plans has a counterpart in the hardware and software plans. The hardware and software plans will be discussed in Subsection 2.6. The plans and schedules in Figure 6 are represented as a hierarchy. For clarity of presentation, the planning documents are connected directly to each other, while the schedules are connected to each other via connectors.

For some sub-topic plans (e.g., integration and verification within the Systems Engineering Management Plan (SEMP) primary topic, and training within the Logistics Management primary topic), two levels of planning are required. The first level provides for planning and guidance for the sub-topic as a whole. For example, the integration sub-topic within the SEMP primary topic will plan for the number of integration events appropriate to the system, and the verification sub-topic will decide which integration events should be used for verification after successful integration. The training sub-topic within the Logistics primary topic will plan the sequence of training courses needed to ensure that user personnel are prepared when the system transitions to operations and maintenance.

The second level provides planning and guidance for a specific instance of the general sub-topic plan and usually is performed later in the life cycle than the overall planning. For example, once the number and content of the integration events have been planned, then, as the time for a specific integration approaches, that individual integration event will be planned in more detail. Once the integration events to be used for verification have been determined, then, as the time for a specific verification event

approaches, that individual verification event will be planned in more detail. Once the number and content of the integration events have been planned, then as the integration event approaches the transition of the individual items needed for that integration would be planned.

2.5.1 Integrated Master Schedule

The IMS documents the networked task-oriented schedule of work required to achieve each IMP significant accomplishment for each program event in each formal agreement. This contains the development master schedule for all organizations participating in the development (i.e., prime contractor, prime-contractor peers, and all subcontractors) either as one large schedule or as many referenced sub-schedules (perhaps maintained by different organizations). All lower-level schedules must be kept consistent with the IMS, and the IMS must reflect accurately those events documented in the IMP.

2.5.2 Systems Engineering Management Plans

Systems engineering is responsible for the system as a whole from the start of development through delivery to the final customer. Its primary focus is to ensure that the system being developed will meet all of the customer's requirements. Systems engineering is the coordinating organization for all specialty engineering disciplines (e.g., reliability/dependability, maintainability, availability, safety, security) at the system level and is responsible for the allocation of requirements (including specialty engineering requirements) to the hardware and software development organizations.

The SEMPs describe the plans for the conduct and technical management of the fully integrated engineering effort needed to define, implement, verify, and deliver the system (or portion of the system) being acquired.

The SEMP describes the approach and methodology needed to translate identified needs (requirements) into design requirements. It also describes the systems engineering processes to be applied to that portion of the system covered by a particular formal agreement. This includes defining the approach and methodology for analyzing missions and environments, and for identifying the system requirements (including design constraints) for implementation, integration, verification, manufacturing, deployment, operations, support, training, and disposal. This also includes the interactions between the systems engineering organization and all other interfacing organizations (e.g., the software organization or a subcontractor's systems engineering organization).

Planning for the systems engineering sub-topics (e.g., reliability, maintainability, supportability, safety, security, integration, verification, and product transition) is included within the systems engineering aggregate topic. Each SEMP sub-topic area describes the processes to be followed for that sub-topic's efforts (including the interactions between the topic-specific organization, such as the reliability organization and all other interfacing organizations), the methods to be used, the approach to be followed for each sub-topic activity, as well as schedules (or references to schedules), personnel, and other resources required for that activity.

For the most part, the sub-topic plans related to the SEMP are contained in separate documents. For example, the system safety plan and security plans usually are developed as separate documents.

2.5.2.1 Integration, Verification, and Transition Sub-Topic Plan

The integration, verification, and transition planning and scheduling for that part of the system governed by a formal agreement is contained in the integration, verification, and transition sub-topic plan of the SEMP primary topic. This plan generally is referred to as the System Integration and Verification Master Plan. The integration of system elements, the verification of requirements, and coordination of the

transition of system hardware and software components between organizations are closely related planning activities and generally are planned together.

Integration planning and scheduling within an organizational group establishes a set of integration events and schedules consistent with other governing system events and schedules (e.g., the events and schedules contained in the IMP and IMS) that, when performed, result in the final product(s) required by that organizational group's formal agreement.

As part of the integration planning, certain hardware and software components may be needed from another organization or may need to be provided to another organization in order to allow an integration to proceed. These items must be planned and scheduled to transition to the integration event on a schedule to support the need. These transitions are coordinated by systems engineering, but the detailed planning for the actual hardware and software transition is contained in the hardware and software transition plans (see Subsection 2.6). Not every integration event requires transition planning, but those that do must include it as part of the planning process.

In concert with the integration and transition planning and scheduling, a number of integration events need to be selected at meaningful points in the integration process that will be used to provide objective verification that the integrated components meet their formally documented requirements. Taken as a whole, the successful completion of the planned and scheduled verification events will result in the acceptance of the system or portion of a system by the acquiring organization. Because of the need to perform the verification events in a formal and controlled environment before independent observers, the verification plan, descriptions, and reports associated with each verification event are documented independently of its associated integration event. Verification plans, descriptions, and reports are discussed in detail beginning in Subsection 3.6. Even though the verification events are performed independently of any associated integration and transition events, it is critical to program success that the System Integration and Verification Master Plan correctly and consistently relate the integration, verification, and transition of system elements.

2.5.2.2 System Integration and Verification Master Plan

The following discussion references Figure 11, which describes the System Integration and Verification Master Plan and associated relationships. See also Subsection 2.3.3. Where appropriate to the discussion, associations between the text and different portions of the figure will be made using corresponding numbers in each.

The SEMP contains the planning for the overall systems engineering effort for the development of the system or portion of the system ❶. The systems engineering schedules contain the integrated set of scheduling for the systems engineering effort consistent with all governing and peer schedules ❶. The System Integration and Verification Master Plan contains the results of the integration, verification, and transition planning ❶.

This Master Plan defines the number of system increments needed to integrate and verify the system ❷ and their contents ❸. The system integration contents consist of the specific versions of the system hardware and software components that will be integrated together to create that system increment ❹. The software components relate to software builds scheduled to support system integration. Software builds contained in the system items (e.g., System Item 1 Version 1) are discussed in Subsections 2.6.1.1 and 2.6.1.2. The System Integration and Verification Master Plan maintains the relationships between these components and other events so that as program changes that affect them occur, effective re-planning can be performed. See Subsection 3.9 for additional information on maintaining relationships between critical information. As discussed in Subsection 2.3, not every integration event has an associated verification event or transition event ❺. However, those integration events that do have

verification or transition events associated with them are identified in the System Integration and Verification Master Plan.

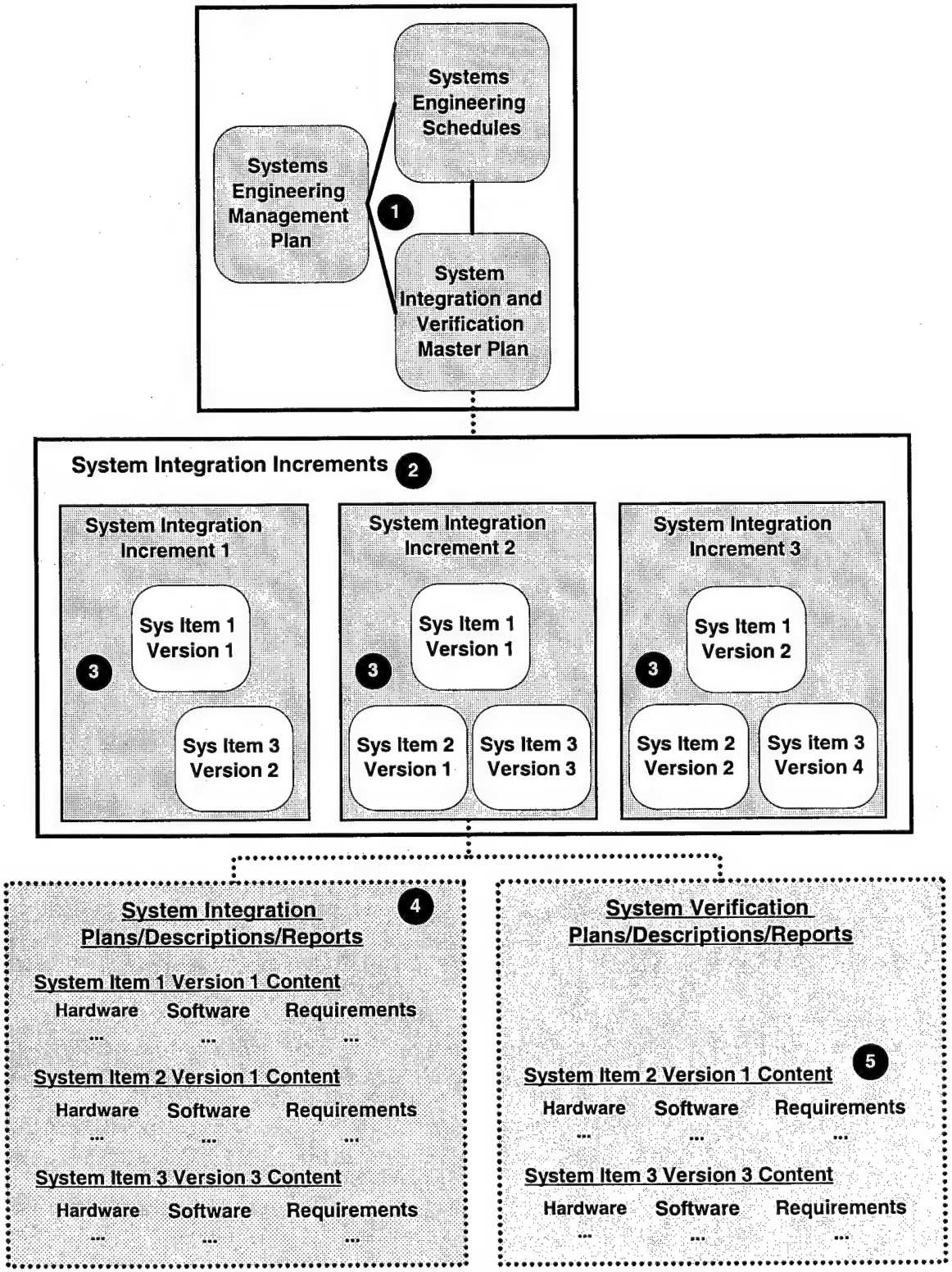


Figure 11. System integration and verification master plan and associated relationships.

2.5.3 Logistics Management Plans

For this report, the planning and scheduling associated with the Logistics Management Plans encompass planning for Logistic Support and Training. In general, these sub-topics are planned, scheduled, and documented independently of each other and are performed by different sub-organizations within a given organizational group.

2.5.3.1 Logistics Support Plan

Logistics is responsible for ensuring that the system being developed will be operable and supportable (maintainable) in its intended environment while meeting the system supportability requirements (i.e., reliability/dependability, maintainability, and availability) and using the personnel and logistics resources available.

Systems engineering is responsible for the actual development of the system and designing in the characteristics to ensure that the system meets its supportability requirements. However, the logistics organization is responsible for ensuring that the system can be supported efficiently and effectively following delivery.

Logistics support planning incorporates a systematic and comprehensive analysis conducted on an iterative basis throughout the development life cycle as an integral part of the systems engineering process. The logistics processes include:

a. Continuous assessment of the analysis results and feedback into the system design to allow for appropriate adjustments to logistics planning or system design as appropriate

b. Procedures for the documentation and monitoring of identified corrective actions

c. Assessments, validations, and verifications to demonstrate that the supportability needs will be met

The goal of logistics planning is to optimize the support system needed to achieve the best balance among cost, schedule, performance, and supportability.

A key to the efficient and effective supportability of any system is (a) how that system handles system failures, (b) how system failures are isolated and the system restored to operations, and (c) how the system components causing system failures are repaired. One of the more serious and frequent deficiencies in system design occurs when the reliability, maintainability, and availability requirements are not coordinated effectively as a whole during system development among systems engineering, logistics, hardware, and software. This deficiency leads to systems that are designed as if the hardware and software were standalone entities that do not have meaningful interactions that affect system supportability. It also leads to systems that do not have a well-designed integrated diagnostics capability that works together across both the hardware and software domains. It is critical for system operations and maintenance that the system design treats hardware and software as integrated components of the system, provides efficient system problem isolation and recovery capabilities following system failures, and provides efficient isolation and repair capabilities for the hardware failures and software faults that occur.

2.5.3.2 Training Plan

The training plan focuses on planning to define the post-delivery training requirements needed by the operations and maintenance personnel, taking into consideration the operation and maintenance policies of the program being developed. This responsibility includes ensuring that adequate training material, facilities, and equipment are available in a timely manner.

Depending on the system requirements, systems engineering may be responsible for the actual development of the training system. However, the training organization is responsible for ensuring that the training system is efficient and effective for training needs.

Training planning incorporates a systematic and comprehensive analysis conducted on an iterative basis throughout the development life cycle as an integral part of the systems engineering process. The training processes include:

a. Continuous assessment of the analysis results and feedback into the system design to allow for appropriate adjustments to training planning or system design as appropriate

b. Procedures for the documentation and monitoring of identified corrective actions

c. Assessments, validations, and verifications to demonstrate that the training needs will be met

The goal of training planning is to optimize the training system needed to achieve the best balance among cost, schedule, performance, and training.

The training analysis includes a full analysis across all operations and maintenance positions and all of the hardware and software associated with those positions. This analysis should be independent of who actually will perform the operations and maintenance roles. In general, the training needs for operations and maintenance personnel are determined in parallel with system development and require close cooperation between systems engineering, hardware and software development personnel, and training, operations, and maintenance personnel.

Planning for the training of development personnel on the system/software engineering environment generally is performed as part of the systems engineering and software development planning and is documented in the SEMP and SDP.

2.5.4 Other Related Plans

For this report, the planning and scheduling associated with other related plans encompass Quality Assurance plans and Configuration Management plans.

2.5.4.1 Quality Assurance Plan

Quality assurance (QA) is intended to act as an independent oversight agent. The QA plan contains planning to ensure that the system is being developed according to its approved processes and procedures, that the products produced conform to their approved standards, that product nonconformance is documented and corrective action taken by the responsible parties, and that written records are maintained. In general, QA personnel focus on either hardware or software. The QA planning generally is contained in two independent documents, the QA Plan for hardware and SQA Plan for software. These documents are developed by QA personnel and may be included in the Hardware Development Plan (HDP) or SDP, either directly or by reference.

Understanding the inherent limitations of the QA processes is essential for any development. The QA processes focus on ensuring that processes and standards are followed and deviations from those processes documented and corrected. The QA processes do not ensure the quality of the content of the work produced by program processes. Only qualified technical personnel knowledgeable in a given field can determine content quality. It is essential to program success that processes are incorporated in the planning for the technical disciplines (e.g., systems engineering, software development, reliability) that evaluate the quality of the content of the work products produced by those processes and that ensure appropriate corrective actions are taken for all identified deficiencies. An example of such a process is the software peer review process.

2.5.4.2 Configuration Management Plan

Configuration Management (CM) is responsible for ensuring that all of the documentation, physical media, and physical parts that represent or comprise a configurable hardware or software component are controlled and accountable during development and at delivery. The hardware and software components controlled include the evolving system and development support environments (engineering, implementation, and test) used to manage, implement, and verify the system.

"Formal" CM focuses on those components and support environments that have been designated by program management for CM control separate from the development and integration organizations (e.g., components or support environments that are deliverable to an external party or are used for verification of a deliverable component). "Informal" CM focuses on those components and support environments either prior to being transitioned into the "formal" CM system or those components and support environments that are designated for CM control by the development or integration organizations.

"Informal" CM is planned for in the HDP and SDP as part of the development process, while "Formal" CM is planned for in the hardware and software CM plans. The hardware and software CM plans may be incorporated in the HDP and SDP, either directly or by reference. For example, "informal" CM plans could be used to control the hardware and software used for integration, while "formal" CM plans could be used to control the integrated hardware and software for verification.

The use of the terms "informal" and "formal" should not be construed to indicate that one is necessarily more rigorous than the other. The degree of configuration control and the rigor of its application should be guided by what the impact to the program would be should a loss of configuration occur.

Some form of configuration management should be in place at every level of development for the system, from the individual developer to the final system to be delivered. Because the planning for the "informal" and "formal" CM processes is performed and documented by different organizations, there is a tendency to use very different processes and procedures for each. The danger in this approach is that the CM processes at each program level must interface with each other in order to transition the hardware and software along as system components are developed and combined and the final system is delivered. For example, the developer's informal CM system must interface with the informal development integration CM system which must interface with the informal integration CM system which must interface with the formal verification CM system. These essential interfaces between the CM systems must be planned just as the interfaces in the system to be delivered must be planned. It is essential for efficient and effective development that these CM systems are compatible and provide the same type of configuration information. This is especially critical across organizational boundaries. The CM planning, both "formal" and "informal" should be performed to ensure integrated CM processes and environments.

2.6 Development and Transition Plans

Hardware and software development plans provide the planning needed to develop the hardware and software components of the system architectural design. Transition plans provide the planning needed to transfer the responsibility and documentation associated with an individual increment of the system hardware and software components from one organization to another.

As discussed in Subsection 2.3.2 and shown in Figures 5 and 6, the HDPs and SDPs are subordinate to the system plans (i.e., the SEMP, Logistics Management Plan, and "Other" Management Plans). Although the HDPs and SDPs are subordinate to the system plans and each organizational group has essentially the same hierarchy of planning and scheduling documents, the scope of the required hardware and software planning and scheduling will differ depending on which organization is performing the planning and scheduling. For example, the hardware and software planning and scheduling done by the prime contractor must encompass the hardware and software for the entire system, including all peer

organizations and subcontractors, as well as its own direct hardware and software development responsibilities, while the planning and scheduling performed by a subcontractor encompasses only its portion of the hardware and software. However, the subcontractor hardware and software planning and scheduling must be consistent with the prime contractor's planning and scheduling.

Within an organizational group, there may be multiple sub-organizations that are responsible for the development of the hardware and software components of a system or portion of a system. For example, in the representative system (Subsection 2.3), the Program Office is responsible for the development of the satellite communication software and the ground segment communication software. The representative system assumes that different sub-organizations within the Program Office will be responsible for each. Because of differences in the hardware and software development needs, each sub-organization may have a different HDP and SDP. The only critical aspect of these HDPs and SDPs is that they are consistent with their governing system plans and among themselves.

An HDP or SDP describes the plans for the conduct and technical management of the fully integrated hardware or software development effort for the hardware or software components allocated to it. All hardware and software to be developed/procured/reused must be allocated to (must follow the processes and procedures in) one HDP or SDP. In some cases, when there are aspects of an organizational group's hardware or software development that must be followed for all hardware or software development (e.g., the use of the object-oriented analysis and development methodology, or common reporting processes), there may be a hierarchy of HDPs and SDPs with a higher level containing the information common to those lower in the hierarchy. This hierarchy of planning information may exist in separate and distinct documents or may be inserted into a single document.

2.6.1 Software Development Plan

The processes and procedures portions of the SDP constitute the "qualitative" planning information needed to execute the software development. This "qualitative" planning information usually has multiple levels. All levels above the bottom level are the processes and procedures that are applicable to multiple organizations/components. These may be imposed at each level between the actual development organization and the prime contractor. The bottom level contains the processes and procedures unique to each organization/software component. For example, the prime contractor may require a common problem reporting process that applies to all software everywhere, a subcontractor may impose an object-oriented development methodology on all software being developed by its organizations, and each subcontractor organization developing software may impose specific coding standards for its software development.

The personnel estimates, staffing profile development, assignment of personnel to properly ordered tasks of appropriate duration, and the preparation of an appropriate infrastructure of hardware and software for the personnel to use constitute the "quantitative" planning information needed to execute the software development. This planning is unique for each organization/software component but must be consistent with both higher-level plans and peer plans. Both qualitative and quantitative planning are required if the development is to be successful.

Each SDP includes relevant sub-topics that correspond to the sub-topics in the system plans (i.e., SEMP, Logistics Management Plan, and "Other" Management Plans). For example, the software development plan(s) will include or reference the software-related planning for the reliability, training, configuration management, and quality assurance sub-topics of the system-level aggregates.

2.6.1.1 Integration, Verification, and Transition Sub-topic Plan

The integration, verification, and transition planning and scheduling for that portion of the software associated with a given SDP is contained in the integration, verification, and transition sub-topic plan of that SDP. This plan generally is referred to as the Software Master Build[12] Plan. The integration of software (computer resource) components, the verification of requirements, and the transition of computer resource hardware and software components between organizations are closely related planning activities and generally are planned together.

Integration planning and scheduling for the software development associated with an SDP establishes a set of integration events and schedules consistent with other governing events and schedules (e.g., the events and schedules contained in the IMP and IMS) that, when performed, result in the final software product(s).

As part of the integration planning, certain computer resources hardware and software components may be needed from another organization or may need to be provided to another organization in order to allow an integration to proceed. These items must be planned and scheduled to transition to the integration event on a schedule to support the need. These transitions are coordinated by systems engineering, but the detailed planning for a specific hardware and software transition are contained in the hardware and software transition plans (see Subsection 2.6.2). Not every integration event requires transition planning, but those that do must include it as part of the planning process.

In concert with the integration and transition planning and scheduling, a number of integration events need to be selected at meaningful points in the integration process that will be used to provide objective verification that the integrated components meet their formally documented requirements. Taken as a whole, the successful completion of the planned and scheduled verification events will result in the acceptance of the system or portion of a system by the acquiring organization. Because of the need to perform the verification events in a formal and controlled environment before independent observers, the verification plan, descriptions, and reports associated with each verification event are documented independently of its associated integration event. Verification plans, descriptions, and reports are discussed in detail beginning in Subsection 3.6. Even though the verification events are performed independently of any associated integration and transition events, it is critical to program success that the Software Master Build Plan correctly and consistently relate the integration, verification, and transition of software (computer resource) components.

2.6.1.2 Software Master Build Plan

The following discussion references Figure 12, which depicts the Software Master Build Plan and its associated relationships. Where appropriate to the discussion, associations between the text and different portions of the figure will be made using corresponding numbers in each.

The SDP contains the planning for the overall software effort for the development of the computer resources associated with it ❶. The software development schedules contain the integrated set of scheduling for the software development effort consistent with all governing and peer schedules ❶. This planning must be consistent with the system-level planning as discussed in Subsections 2.5.2.1 and 2.5.2.2 and shown in Figure 11. The relationship between software integration builds and system integration increments is illustrated in Subsection 2.3.3.

[12] The term "Build" is a common term for software that refers to the set of software to be integrated together to provide some "agreed to" capability.

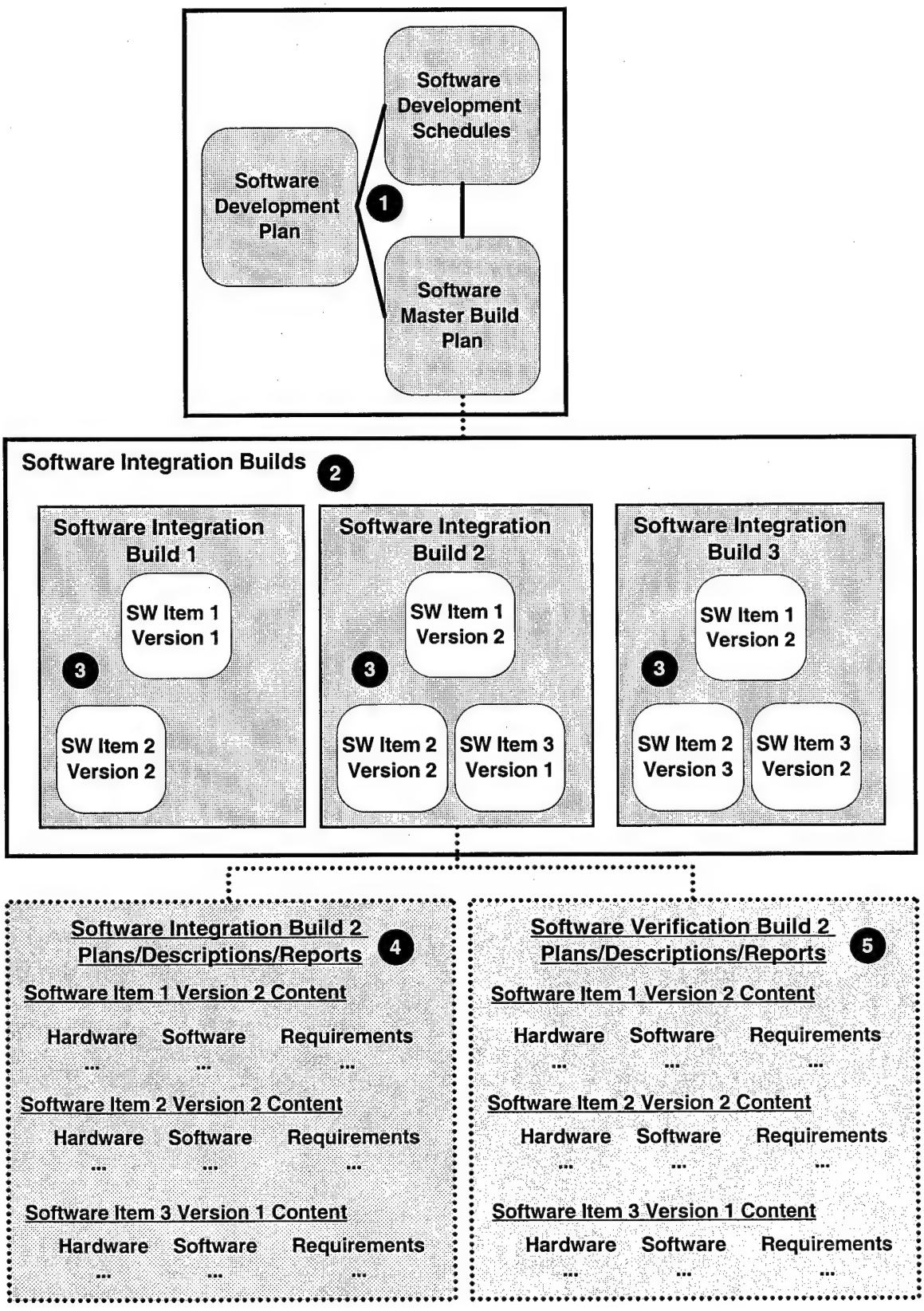


Figure 12. Master build plan and associated relationships.

The Software Master Build Plan contains the results of the integration, verification, and transition planning ❶. This Plan defines the number of software integration builds needed to develop and verify the software ❷ and their contents ❸. The software integration contents consist of the specific versions of the computer resource hardware and software components that will be integrated together to create that software increment ❹. The Software Master Build Plan maintains the relationships between these components and other events so that as program changes that affect them occur, effective re-planning can be performed. See Subsection 3.9 for additional information on maintaining relationships between critical information. As discussed in Subsection 2.3, not every integration event has an associated verification event or transition event ❺. However, those integration events that do have verification or transition events associated with them are identified in the Software Master Build Plan.

2.6.2 Software Transition Plan

Software transition encompasses the planning that provides an orderly and documented transfer of software from one organization to another. Software transitions may occur for many purposes, e.g., integration testing, verification, delivery to operations, and maintenance. Software transitions occur at intervals planned in the SDP throughout the development life cycle of a system. Transitions can occur within an organization; for example, the transition within a software development organization from the developer to the development lead following unit verification or the transition from the software development organization to the integration organization following initial internal software integration. Transition also can occur between organizational groups, for example, the transition between a subcontractor and the prime contractor, between a peer organization and the prime contractor, or between the prime contractor and the acquiring organization following the completion of an increment of work. Transition to the acquiring organization actually may result in the transition of the work increment (or final product) to the appropriate operations and maintenance organizations for which the development effort was performed.

For the purposes of this report, it is assumed that software transitions that are within organizational groups are planned as part of integration planning for that group and that planning for the software transition between organizational groups will be included in a separate Software Transition Plan (STrP).

In all cases, the information needed to execute a transition is similar. The hardware, software, facilities, documentation (e.g., requirements documents and architectural descriptions), personnel, and other resources (e.g., consumables) needed by the software being transitioned, as well as the specific transition steps that must be performed in order to transfer the software and verify that the transition was successful, must be documented.

In addition to the information defined above, the information called out for the Software Product Specification (SPS), (see Subsection 4.3.2), and the Software Version Description, (see Subsection 4.3.3), also is required. For less formal or less complex transitions between organizational groups, this information could be included directly in the transition plan. However, for the more formal or complex software transitions (e.g., between the prime contractor and the acquiring organization), this information along with all of the other information in Section 4 should be documented separately.

If training is required for the personnel receiving the software, it is included in the training plan described in Subsection 2.5.3.2.

If multiple transitions/installations are required at different times (e.g., a test facility for testing, followed later by installation at one or more other sites), then multiple STrPs are needed. Planning for the entire set of software installations and transitions over multiple time periods is contained in the SDP, while the planning for each installation and transition at each specific time period is contained in the STrP.

The STrP assumes that all computer resource hardware and non-computer resource hardware needed by the software has been preinstalled as planned in the Hardware Transition Plan (HTrP).

2.6.3 Hardware Development Plan

The processes and procedure portion of the HDP constitute the "qualitative" planning information needed to execute the hardware development. This "qualitative" planning information usually has multiple levels. All levels above the bottom level are the processes and procedures that are applicable to multiple organizations/components. These may be imposed at each level between the actual development organization and the prime contractor. The bottom level contains the processes and procedures unique to each organization/hardware component. For example, the prime contractor may require a common problem reporting process that applies to all hardware everywhere, a subcontractor may impose the use of specific engineering drawing methodology on all hardware being developed by its organizations, and each subcontractor organization developing hardware may impose specific testing standards for its hardware development.

The personnel estimates, staffing profile development, assignment of personnel to properly ordered tasks of appropriate duration, and the preparation of an appropriate infrastructure of hardware and software for the personnel to use constitute the "quantitative" planning information needed to execute the hardware development. This planning is unique for each organization/hardware component but must be consistent with both higher-level plans and peer plans. Both qualitative and quantitative planning are required if the development is to be successful.

Each HDP includes relevant sub-topics that correspond to the sub-topics in the system plans (i.e., SEMP, Logistics Management Plan, and "Other" Management Plans). For example, the hardware development plan(s) will include or reference the hardware-related planning for the reliability, training, configuration management, and quality assurance sub-topics of the system-level aggregates.

In many cases, separate HDPs are not used to document the hardware development planning. Instead, the processes and procedures needed for hardware development are included directly in the SEMP.

Currently, there is no appropriate existing DID that can be used as the basis for tailoring for the HDP.

2.6.3.1 Integration, Verification, and Transition Sub-topic Plan

The integration, verification, and transition planning and scheduling for that portion of the hardware associated with a given HDP is contained in the integration, verification, and transition sub-topic plan of the HDP. This plan generally is referred to as the Hardware Integration and Verification Master Plan. The integration of hardware elements, the verification of requirements, and the transition of hardware elements between organizations are closely related planning activities and generally are planned together.

Integration planning and scheduling for the hardware development associated with an HDP establishes a set of integration events and schedules consistent with other governing events and schedules (e.g., the events and schedules contained in the SEMP) that, when performed, result in the final hardware product(s).

As part of the integration planning, certain hardware items may be needed from another organization or may need to be provided to another organization in order to allow an integration to proceed. These items must be planned and scheduled to transition to the integration event on a schedule to support the need. These transitions are coordinated by systems engineering, but the detailed planning for the actual hardware transition is contained in the hardware transition plans (see Subsection 2.6.4). Not every integration event requires transition planning, but those that do must include it as part of the planning process.

In concert with the integration and transition planning and scheduling, a number of integration events need to be selected at meaningful points in the integration process that will be used to provide objective verification that the integrated elements meet their formally documented requirements. Taken as a whole, the successful completion of the planned and scheduled verification events will result in the acceptance of the system or portion of a system by the acquiring organization. Because of the need to perform the verification events in a formal and controlled environment before independent observers, the verification plan, descriptions, and reports associated with each verification event are documented independently of its associated integration event. Verification plans, descriptions, and reports are discussed in detail beginning in Subsection 3.6. Even though the verification events are performed independently of any associated integration and transition events, it is critical to program success that the Hardware Integration and Verification Master Plan correctly and consistently relate the integration, verification, and transition of hardware elements.

2.6.3.2 Hardware Integration and Verification Master Plan

The HDP Hardware Integration and Verification Master Plan has a similar relationship structure as that shown in Figure 11.

2.6.4 Hardware Transition Plan

Hardware transition encompasses the planning that provides an orderly and documented transfer of hardware from one organization to another. Hardware transitions may occur for many purposes, e.g., integration testing, verification, delivery to operations, and maintenance. Hardware transitions occur at intervals planned in the HDP (or SEMP if no HDP is used) throughout the development life cycle of a system. Transitions can occur within an organization, for example, the transition within a hardware development organization from the hardware item developer to the development lead following item verification or the transition from the hardware development organization to the integration organization following initial internal hardware integration. Transition also can occur between organizational groups, for example, the transition between a subcontractor and the prime contractor, between a peer organization and the prime contractor, or between the prime contractor and the acquiring organization following the completion of an increment of work. Transition to the acquiring organization actually may result in the transition of the work increment (or final product) to the appropriate operations and maintenance organizations for which the development effort was performed.

For the purposes of this report, it is assumed that hardware transitions that are within organizational groups are planned as part of integration planning for that group, and that planning for the hardware transition between organizational groups will be included in a separate HTrP.

In all cases, the information needed to execute a transition is similar. The hardware, software, facilities, documentation (e.g., requirements documents and architectural descriptions), personnel, and other resources (e.g., consumable items) needed by the hardware being transitioned, as well as the specific transition steps that must be performed in order to transfer the hardware and verify the transition was successful, must be documented.

In addition to the information defined above, the information called out for the Hardware Product Specification (Subsection 4.3.6) also is required. For less formal or less complex transitions between organizational groups, this information could be included directly in the transition plan. However, for the more formal or complex hardware transitions (e.g., between the prime contractor and the acquiring organization), this information along with all of the other information in Section 4, should be documented separately.

If training is required for the personnel receiving the hardware, it is included in the training plan described in Subsection 2.5.3.2.

If multiple transitions/installations are required at different times (e.g., a test facility for testing, followed later by installation at one or more other sites), then multiple HTrPs are needed. The planning for the entire set of hardware installations and transitions over multiple time periods is contained in the HDP, while the planning for each installation and transition at each specific time period is contained in the HTrP.

The HTrP assumes that all computer resource hardware and non-computer resource hardware is planned in the HTrP. The HTrP also assumes that all software needed by the hardware to perform its intended functions is installed as planned in the STrP.

Currently, there is no appropriate existing DID that can be used as the basis for tailoring for the HTrP.

3. Requirements/Design/Verification Relationships

This section describes the requirements, design, and verification product relationships. It assumes that the "user organization" develops its own system requirements and concept of operations and documents them in some formal documentation. For the purposes of this report, the user's requirements are documented in the Operational Requirements Document (ORD). It assumes that the "acquisition organization" develops an abbreviated set of encompassing performance-related requirements and documents them in some formal documentation. For the purposes of this report, the acquisition organization's requirements are documented in the "System Performance Document (SPD)." It further assumes that a "Development Organization" creates an acquisition product hierarchy for the system to be acquired. This acquisition product hierarchy consists of: (a) a system, segment,[13] and B-level item[14] specification hierarchy containing the system's requirements (known as the specification tree); (b) a design product hierarchy containing the system's architectural description and design documents that relate to the specification tree and containing the system's design; and (c) a verification product hierarchy that also relates to the specification tree and contains the system's specific verification plans, descriptions, and reports. The text in the subsequent paragraphs of this section refers specifically to elements in Figure 13.

The documentation of the systems engineering, system verification, hardware development, and software development processes are discussed in Section 2. The execution of these documented processes results in the definition of the specification, design, and verification documentation trees, and the development of the products discussed in this section. Support processes such as configuration management and quality assurance needed for the products discussed herein also are included in Section 2 and assumed here.

The requirements, design, and verification products for the system are shown in three boxes. Requirements products are shown in the middle box, since both design and verification products flow from requirements products. External requirements and verification are shown for completeness but are not intended to be included as part of the system products needed for future tailoring.

As shown in Figure 13, the software acquisition products (identified as "Included in EIA/IEEE J-STD-016" in the figure's legend) are not self-contained within the total set of system products. This is an artifact of the original structure of the software-centered philosophy embodied in MIL-STD-498 (Ref 1) and its predecessor standards. The implication of this is that considerable additional work at the system level is required to ensure a consistent set of products for a specific program.

---

[13] The term "segment" rather than "subsystem" (used by the EIA/IEEE J-STD-016-1995) has been used consistently throughout this report, because "segment" is the more common term for the described entity in the SMC/NRO and associated contractor community.

[14] For clarity, the term "B-level item" as used here encompasses all hardware, combined hardware and software, and software configuration items contained in the specification tree below the level of the segment(s). The term specifically includes Software Requirement and Interface Specifications.

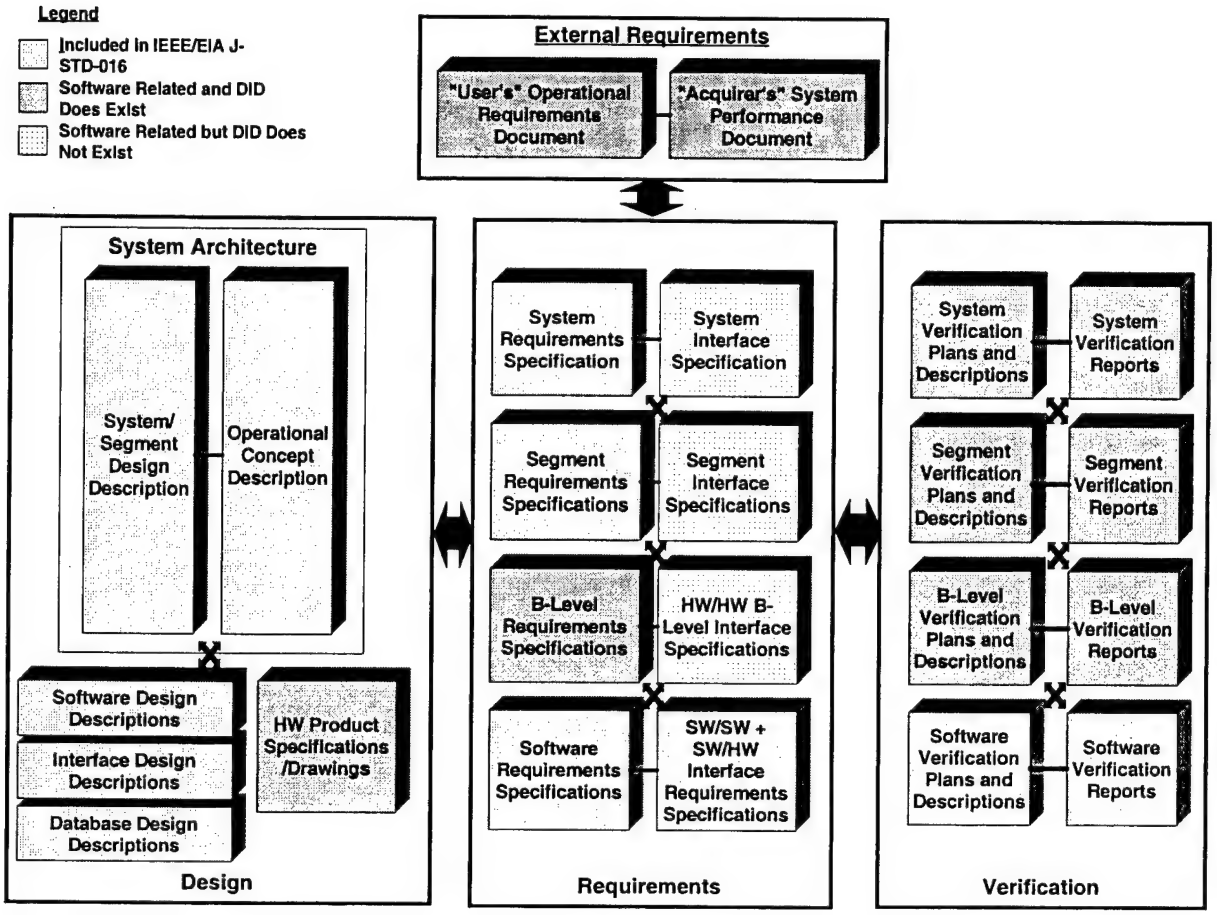


Figure 13. Requirements/design/verification relationships.

Within the "Requirements" box, system, segment, and hardware interface specification DIDs comparable to the software Interface Requirements Specification (IRS) DID do not exist and will need to be developed. Specifying interface requirements between entities in a separate and independent interface specification provides a significant advantage for control and consistency over including those interface requirements within the specification of each entity that uses the interface. This model of using separate interface specifications that are referenced by the specifications of the interfacing entities is assumed throughout the remainder of this report.

Within the "Design" box, the System/Segment Design Description (SSDD) and the Operations Concept Description (OCD) provide the mechanism to describe and maintain the total system architecture (including computer resources[15]) and the philosophy of operation for the life of the system (both development and maintenance).

The SSDD also provides the primary reference(s) to the requirements, allocation, and verification traceability repository. This repository (not shown in the figure) consists of one or more physical information structures which provide system-wide bidirectional traceability for all system requirements (including interface requirements), allocable items (e.g., performance timelines and reliability), and

[15] The term "computer resources" rather than "software" has been used consistently throughout to emphasize the fact that software never stands alone outside the context of the hardware system on which it operates. This is true even when the hardware is a commercial product.

requirements verification records (e.g., verification methods, levels, procedures, and status of each requirement) that are to be maintained for the life of the system.

Previous experience with less comprehensive traceability methodologies resulted in both the developer and the acquirer being unable to unambiguously determine whether all of the requirements were being allocated and implemented appropriately during development and whether all of the requirements were being verified appropriately during the formal verification process. The need for a central repository of this information with the ability to access the appropriate information as needed for program use (e.g., automatic insertion of requirements traceability into specific documents directly from the central repository) is a direct result of this experience. The existence of this traceability repository is assumed throughout the remainder of this report. The characteristics of this repository will be discussed in Subsection 3.9.

Within the "Verification" box, the verification plans, verification descriptions, and verification reports are shown at each level of the system verification hierarchy. The DIDs for the System, Segment, and Non-software B-level Verification Plans, Descriptions, and Reports comparable to the software DIDs do not exist and will need to be developed. The generalization of the software DID approach to verification for the entire specification tree provides a significant advantage by providing a consistent verification documentation hierarchy for the entire system. This model for the organization of the verification approach and associated products is assumed throughout the remainder of this report.

3.1 External Requirements

3.1.1 "User's" Operational Requirements Document

The using organization generally provides an ORD to define the system to be acquired. The ORD contains a concept of operations (CONOPS) for the system to be procured. This CONOPS describes how the user expects the system to operate in the fielded environment. In non-requirements language, the CONOPS description explains the user's expectations for the system in all of its critical operational environments and workloads. This description provides the conceptual framework to be used to help interpret the actual requirements that must be satisfied by the system. The ORD also provides a formal set of requirements that the system must meet in order to be accepted for operations. The acquisition organization is responsible for ensuring that the final system meets the user requirements.

3.1.2 "Acquirer's" System Performance Document

The acquisition organization generally provides an SPD to define the performance-specific requirements that the system to be procured must meet. This document forms the contractual basis from which the System Specification is derived by the development organization.

3.2 System Requirements and Design Documentation Introduction

Each acquisition requires an acquisition-specific hierarchy of specifications and design products. This product hierarchy defines the decomposition of the system to be acquired into its constituent items and describes the items' relationships and operation. In order to describe the content and relationships among the specification and design product types, a representative specification tree has been defined in Subsection 3.3. This representative specification and design tree contains at least one of each type of specification and design product normally used by SMC/NRO acquisitions. These figures will be referenced as necessary to support the individual product discussions below. In order for these products to be used for a specific acquisition, the relationships and content described herein must be appropriately translated into that acquisition's documentation tree.

Requirement specifications describe what the system is supposed to do and how well it is supposed to do it. Requirement documents are discussed in Subsection 3.4. Architecture design descriptions explain how the system items relate to each other as a whole and how the system operation will meet the user's concept of operations. Design descriptions explain how the individual items in a system are constructed such that the items meet the stated requirements. Design documents are discussed beginning in Subsection 3.5. Requirement specifications also contain the qualification requirements for the portion of the system covered by the specification. These qualification requirements define what verification methods are to be used to determine that the stated requirements are met and at what level in system integration the methods will be applied. Verification documents are discussed beginning with Subsection 3.6.

3.3 Representative Specification and Design Documentation Tree

This section illustrates the relationships among the various requirements, design, and verification products in the context of a representative system. The specification and design documentation tree of a portion of the representative system is shown in Figures 14 through 16. This representative documentation tree provides a framework that can be used to apply the appropriate relationships to a specific system.

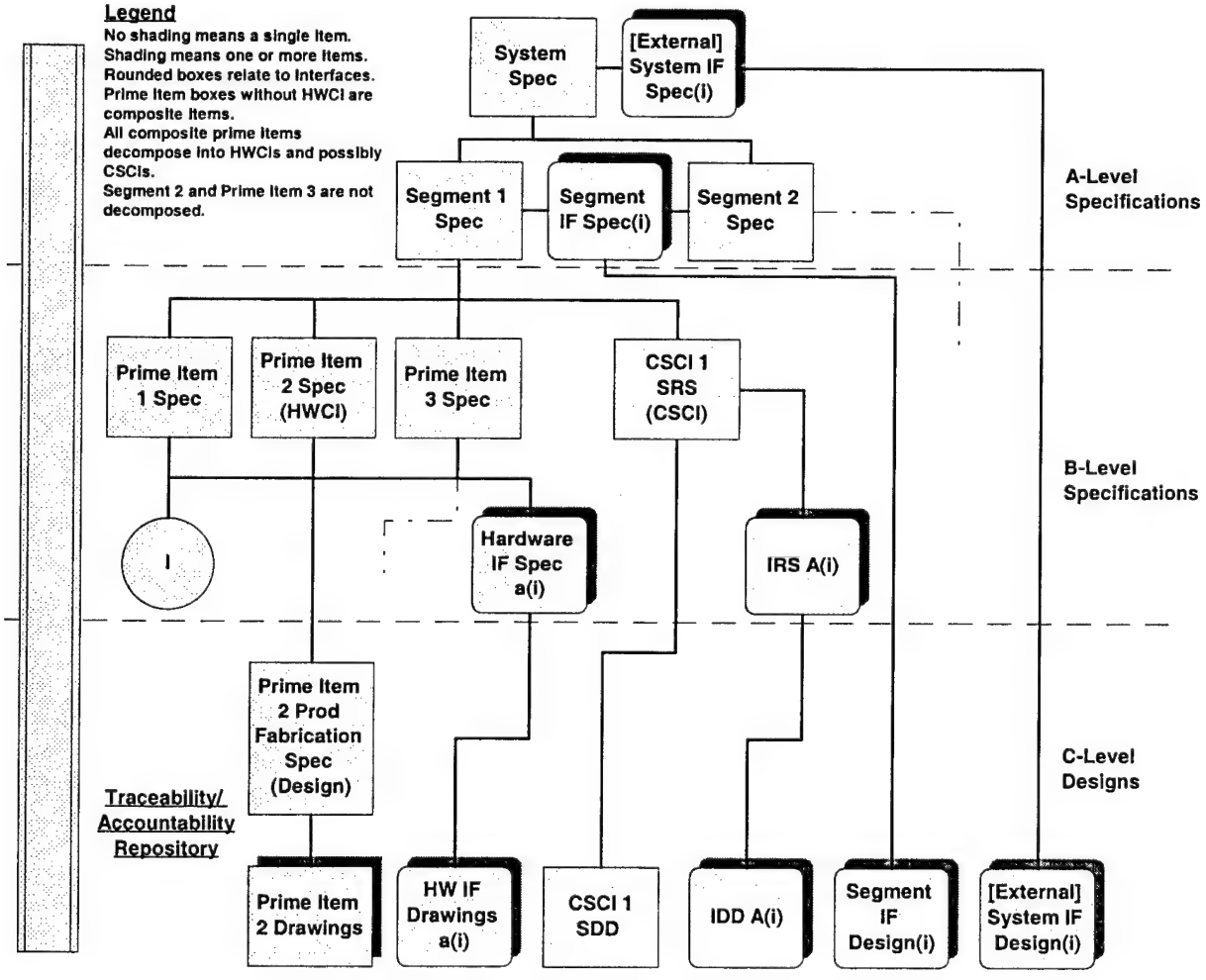


Figure 14. Representative specification/design documentation tree (1 of 3).

The representative system comprises two segments (e.g., a system consisting of one or more satellites and one or more ground stations), Segments 1 and 2 (see Figure 14). The system and segment levels also are called the A-levels in reference to the type of specification content/format used to document the requirements. In general, the system and segment requirements specifications and their associated interface specifications constitute the A-level.[16] The system and segment specifications use the content/format of the System/Segment Specification (SSS) for documenting requirements.

Interface requirements[17] at the A-level are documented in System/Segment Interface Specifications (SSISs). The requirements for any specific interface are documented in only one interface specification, chosen appropriately for the system. All interfaces that are external to the system being acquired are documented in SSISs at the system level whether or not they are associated with only one segment. All interfaces between segments, but not external to the system, are documented in SSISs at the segment level.

The interface requirements documented in SSISs may be organized as appropriate to the system as long as the requirements for a specific interface reside in only one SSIS. Since interfaces between elements represent agreements between peers, change control for interface specifications is performed at the lowest system level that has decision-making responsibility and authority for both sides of any interface contained in a given interface specification. For example, change control over both system and segment interface specifications would be performed at the program level (e.g., the Level-1 IPT, or program manager) unless decision-making responsibility and authority is delegated to another program organization (e.g., a specific segment, Level-2 IPT, or System Engineering and Integration Team). See Section 2 for information on the planning for the management of change control.

External and inter-segment interfaces may be hardware to hardware, hardware to software, software to software, or system to human. Currently, there is no appropriate existing DID that can be used as the basis for tailoring for these products.

The only design documents derived from the A-level, other than the system architecture,[18] are the System/Segment Interface Design Descriptions. The interface design documents are derived from their respective SSISs. The design level also is called the C-level in reference to the type of design documentation content/format used to document the design. Currently, there is no appropriate existing DID that can be used as the basis for tailoring for the SSIS.

In the representative system (see Figure 14), Segment 1 consists of four items: three prime items (Prime Items 1-3) and one computer software configuration item (CSCI 1). The levels below the A-level are called the B-level in reference to the type of specification content/formats used to document the requirements. There are three types of specification content/formats typically used to document requirements at the B-level for SMC/NRO systems. These are Prime Item [Development][19] Specifications (PIDSs), Critical Item [Development] Specifications (CIDSs), and Software Requirements Specifications (SRSs). The PIDSs are used to document the requirements for both composite prime items as well as hardware configuration items (HWCIs). A composite prime item is composed of one or more lower-level items. An HWCI is not decomposed into any lower-level items, i.e., it is the lowest level item for a specific branch of the specification tree. Lower-level items below a composite prime item can be

---

16 In some large systems a third level, called an Element, may also exist at the A-level and use the SSS to document requirements.

17 Interface Control Documents (ICDs) have traditionally been used to define all non-software interface information. However these documents never have been structured to effectively make a meaningful distinction between requirements and design. For this reason, this report will not use ICDs.

18 See Subsection 3.5.1 for more information.

19 For simplicity, the specification names in the figures and their associated text do not contain the word "development" as part of their title. Elsewhere in the discussion the full title is used.

more composite prime items, HWCIs or CSCIs. The HWCIs and CSCIs are the lowest level items for which the configuration of a system is controlled and requirements are documented. In the representative system, Prime Item 1 is a composite prime item. Prime Item 2 is an HWCI, and CSCI 1 is a computer software configuration item.

Interface requirements at the B-level for hardware-to-hardware interfaces are documented in Hardware Interface Specifications (HISs). Currently, there is no appropriate existing DID that can be used as the basis for tailoring for the HIS. Interface requirements at the B-level for hardware-to-software and software-to-software interfaces are documented in IRSs. The interface requirements documented in one or more HIS or IRS at any level of the specification tree may be organized as appropriate to the system as long as the requirements for a specific interface are not contained in any interface specification at a higher level (including the A-level) and reside in only one HIS or IRS. For example, if a software-to-software interface were to be from a CSCI within the system to a software item outside the system, the requirements for that interface would be documented in an SSIS at the system level and not in any IRS at the B-level. Since interface requirements are contained in only one HIS or IRS, each specification (HISs and IRSs) at the B-level has a corresponding interface design document at the C-level, even though a related B-level specification may be decomposed into a lower level requirements specification and has no C-level design document associated with it.

In the representative system (see Figure 14), appropriate hardware-to-hardware interfaces between the prime items or between the prime items and lower-level hardware items are documented in the HISs named "HW IF Spec a(i)." Appropriate hardware-to-software and software-to-software interfaces between the prime items, lower-level hardware, and lower-level software items and CSCI 1 are documented in the IRSs named "IRS A(i)." Here and in subsequent figures, the subscripts "x(i)" and "X(i)" in the interface specification or design description names indicate the organization of the interfaces and associated designs into one or more logical groupings as appropriate to the system.

At any given B-level, design documents are developed only from those items that are HWCIs, CSCIs, HISs, or IRSs. For prime items that are HWCIs, such as Prime Item 2 in Figure 14, the associated design products contained in the C-level consist of a Prime Item Product Fabrication Specification[20] and the hardware drawings referenced by the fabrication specification. For CSCIs, such as CSCI 1 in Figure 14, the associated design documents contained in the C-level consist of Software Design Documents (SDDs). For HISs, such as HW IF Spec a(i), the associated design documents contained in the C-level consist of the Hardware Interface Drawings named "HW IF Drawings a(i)." For IRSs, such as IRS A(i), the associated design documents contained in the C-level consist of the Interface Design Descriptions (IDDs) named "IDD A(i)."

In the representative system, Prime Item 1 consists of Prime Item 4 and two CSCIs named "CSCI 2 and 3" (see Figure 15). Prime Item 4 also is a composite prime item consisting of a critical item HWCI named "Critical Item 1," two CSCIs named "CSCI 4 and 5," and a product function HWCI named Product Item 1 (see Figures 15 and 16). In the representative system, appropriate hardware-to-hardware interface requirements are contained in the HISs named "HW IF Spec b(i)," and appropriate hardware-to-software and software-to-software interface requirements are contained in the IRSs named "IRS B(i)."

A composite prime item may be decomposed into one or more critical items when specific portions of a prime item are sufficiently complex or important that they warrant being treated as independent HWCIs. It also may be decomposed into one or more product items when specific portions of a prime item do not require detailed design and hardware drawings to obtain the item. This generally is the case where commercial capabilities (e.g., workstations or local area networks) or other government capabilities (e.g.,

[20] The specifications at the C-level contain design descriptions that define how an item is to be developed, while the specifications at the A- and B-levels contain requirements that state what the item is expected to do.

Global Positioning System receivers) are incorporated into the system or equipment. The design documentation associated with those product items consists of the design documentation provided with commercial equipment or the developing agency.

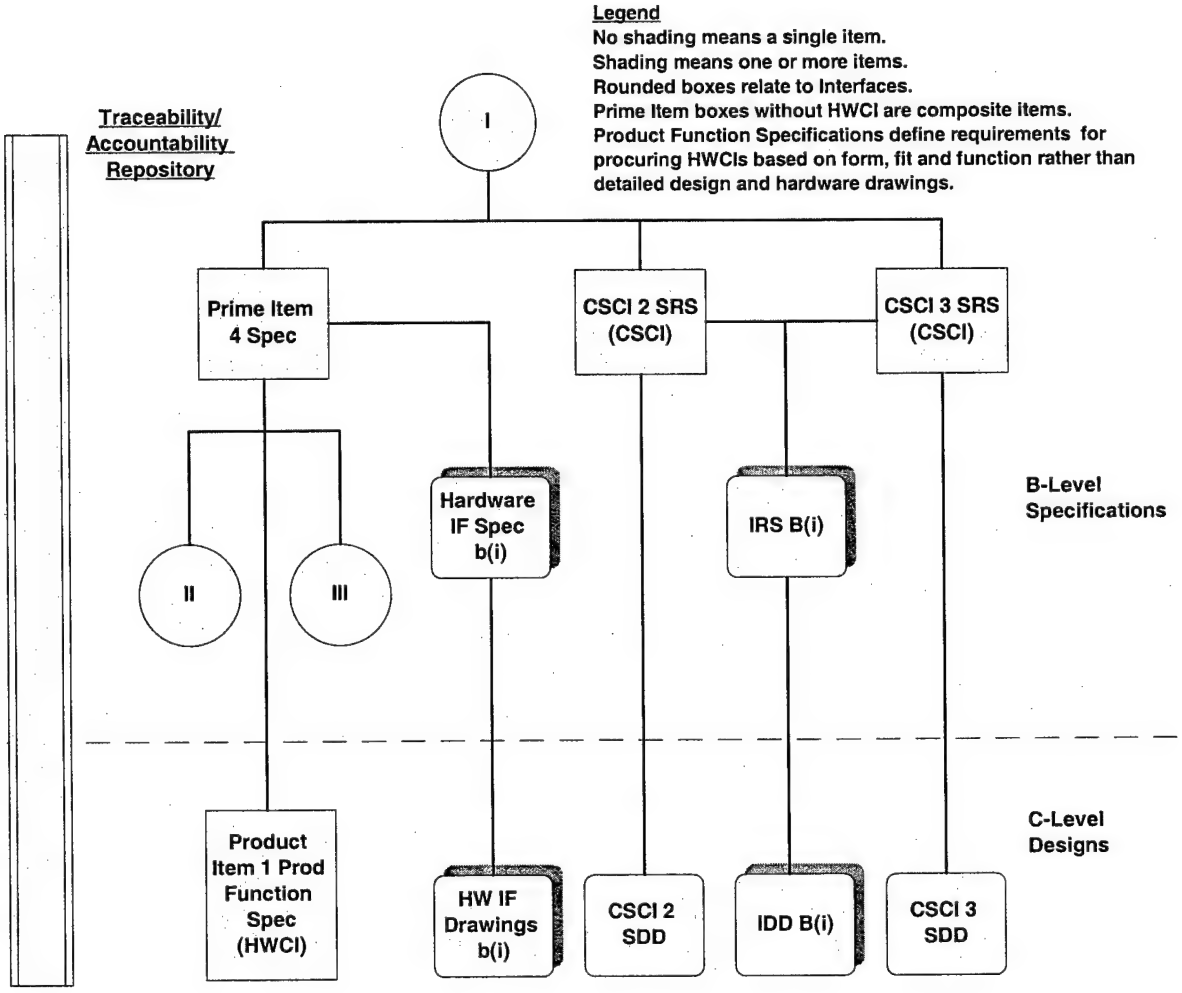


Figure 15. Representative specification/design documentation tree (2 of 3).

As shown in the representative system, Critical Item 1 is an HWCI (see Figure 16). The design for this HWCI is contained in the corresponding critical item product fabrication specification and its associated referenced critical item drawings. In the representative system, the requirements for CSCI 4 include the requirements for a database (see Figure 16). The design for this CSCI is contained in the corresponding database design description named "CSCI 4 DBDD" and software design description named "CSCI 4 SDD." In the representative system, the hardware-to-hardware interface requirements between Critical Item 1 and other hardware CIs (not contained in the system or segment interface specifications or other HISs) are contained in the HISs named "HW IF Spec c(i)." Hardware-to-software and software-to-software interface requirements between CSCI 4 and 5 and other CSCIs in the system (not contained in the system or segment interface specifications or other IRSs) are contained in the IRSs named "IRS C(i)."

The requirements for a database can be included as part of the requirements for a specific CSCI, or the database can be structured separately and independently in its own right with an independent SRS. When it is decided to treat a database as an independent CSCI, then the design documents associated with the

database SRS are only database design descriptions (DBDDs) rather than DBDDs and SDDs as shown in Figure 16.

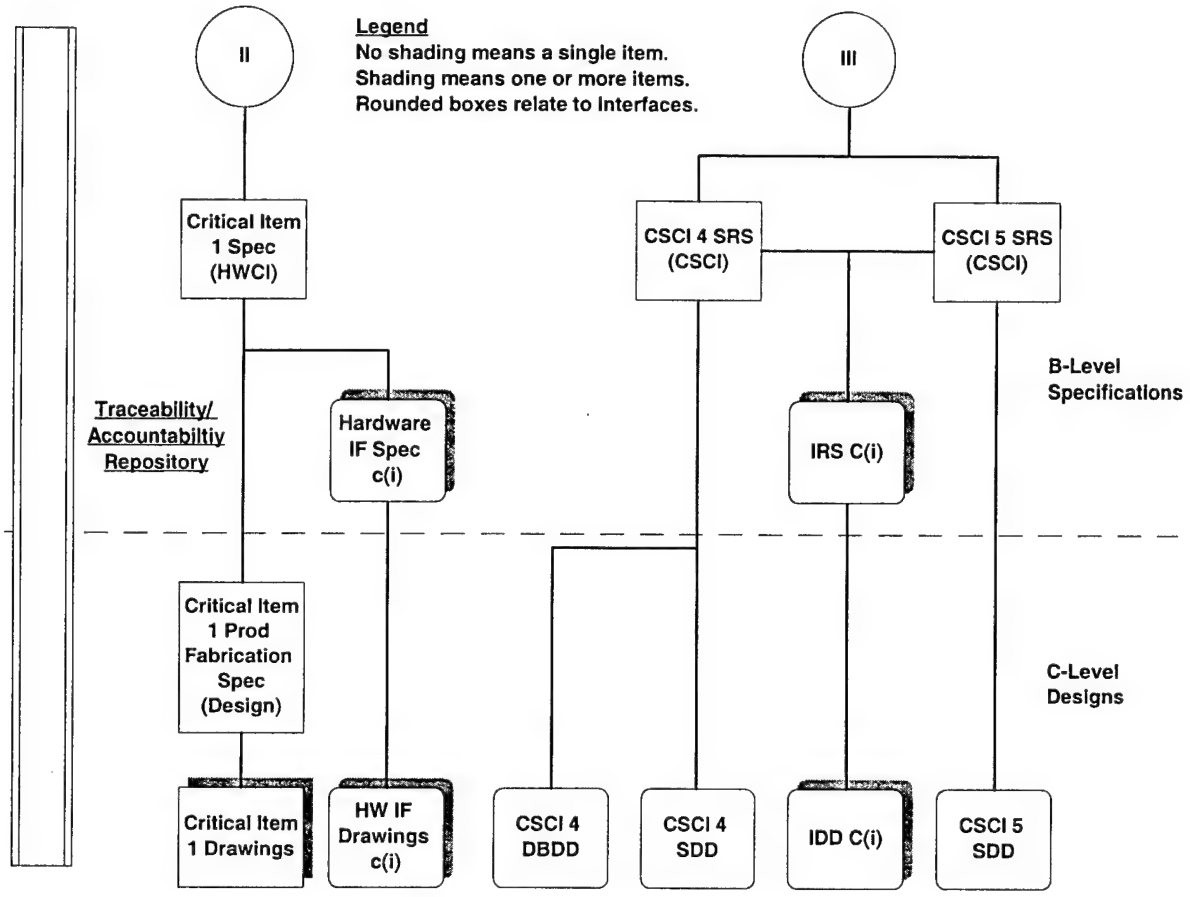


Figure 16. Representative specification/design documentation tree (3 of 3).

For simplicity, Segment 2 and Prime Item 3 in Figure 14 are not decomposed further and are not discussed.

At the left in Figures 14 through 16 is a symbol titled "Traceability/Accountability Repository." This symbol represents one or more information structures that contain the system-wide repository for certain critical information and the bi-directional traceability of certain critical information developed over the life of the system. Although the complete definition of what information needs to be traceable and accountable will be unique to a specific program, a certain minimum set must be included. For the purposes of this report, that minimum set consists of traceability and accountability related to the documentation set contained in Figure 13. The characteristics of the minimum set of traceability and accountability required for any program are described in Subsection 3.9.

3.4 Requirements

Requirements specifications at any level in the specification tree have a common general outline. A narrative introduction provides a context for the specification, followed by the actual requirements for that portion of the system covered by the specification, a qualification section that defines at what level or levels of system integration each requirement will be verified and the verification method or methods to be used to verify each requirement at each level, and bi-directional traceability and accountability for all

requirements in the specification both to the parent specifications and to the child specifications.[21] See Subsection 3.7 for more information on verification, and Subsection 3.9 for more information on traceability and accountability.

As discussed in Subsection 3.3, interface requirements may be organized as appropriate to the system as long as the requirements for a specific interface reside in only one interface specification and all specifications that use a specific interface reference the requirements in the appropriate interface specification.

The requirements specification tree identifies the documents containing the requirements that must be met by the system being developed and provides an outline of the static architectural components that make up the system. However, these documents do not define the interrelationships of the system architecture (i.e., the dynamic interaction of its components) or the operational concept of how the entire system will be organized and will work together to satisfy customer needs. This information is provided by the system architecture design descriptions called the System/Segment Design Description (SSDD) and the Operational Concept Description (OCD). The SSDD and the OCD together describe the complete system architecture and how that architecture operates to satisfy the system requirements. The SSDD and OCD documents must evolve over time as the specification tree is developed and must be maintained consistent with the system requirements throughout system development (see Subsection 3.5.1).

3.4.1 A-level Specifications

The system and segment decomposition generally is defined by the acquiring organization. For example, the acquiring agency might structure the acquisition of a satellite capability into a satellite system consisting of a space segment that includes one or more satellites and a separate ground segment for each ground station location. The system/segment specifications and system/segment interface specifications are created by the development organization based on external requirements provided by the acquiring organization. The system/segment specifications form the performance base for the development and production of lower-level system elements. This set of system/segment specifications and interface specifications also is referred to as the Functional Baseline.

System/segment and interface requirements must be traceable to the acquirer's or user's contractual requirements, or to design decisions documented in the system architecture design description, and must be maintained consistent with those contractual requirements and design decisions. System/segment and interface requirements also must be traceable to the appropriate requirements in lower level (B-level) specifications.

3.4.1.1 System/Segment Specifications

The SSSs state the technical and mission requirements for a system/segment as an entity. They also document design constraints and identify the interfaces to external systems, between segments and possibly between lower-level requirements groupings. They also define the qualification requirements to be used with each requirement to ensure that each A-level requirement has been met. System interfaces must be described in sufficient detail to ensure a clear understanding of the system interface requirements; however, the actual interface requirements are contained in one or more system/segment interface specifications.

---

[21] Specifications at the top of the specification tree have only bi-directional traceability to the child specifications. Specifications at the bottom level of the specification tree contain only bi-directional traceability to the parent specifications. Traceability between the lowest level specification and the design documents that will implement the requirements must be maintained in the traceability repository but will not appear in the specifications themselves.

3.4.1.2 System/Segment Interface Specifications

The SSISs contain the requirements for external and inter-segment interfaces for the system. A given A-level interface specification states the technical requirements for one or more interfaces that are documented together for the specific system to be developed. External and inter-segment interfaces may be hardware-to-hardware, hardware-to-software, software-to-software, and/or system-to-human. Although there is no single existing DID that is appropriate for tailoring for the SSIS, segregating the interface types into three sections [(a) hardware-to-hardware, (b) hardware-to-software and software-to-software, and (c) system-to-human], allows the format/content of the software IRS to be used for items b and c.[22]

3.4.2 B-level Development Specifications

The B-level decomposition generally is defined by the development organization. The B-level development specifications and interface specifications are based on requirement allocations from parent specifications (i.e., specifications at the next higher level in the specification tree, such as the segment specification). The set of B-level specifications and interface specifications also is referred to as the Allocated Baseline.

Three types of specification content/format typically are used to document requirements at the B-level for SMC/NRO systems. These are PIDSs, CIDSs, and SRSs. The PIDSs are used to document the requirements for both composite prime items and HWCIs. A composite prime item is composed of one or more lower-level items. Lower-level items below a composite prime item can be additional composite prime items, HWCIs or CSCIs. The HWCIs and CSCIs are the lowest level items for which the configuration of a system is controlled and requirements are documented, i.e., they are the lowest level items for their specific branch of the specification tree. The CIDSs are used to define the requirements of a hardware subset of a composite prime item that is complex or important enough to warrant a separate specification of its own.

Interface requirements at the B-level for hardware-to-hardware interfaces are documented in HISs. There currently is no appropriate existing DID that can be used as the basis for tailoring for the HIS. Interface requirements at the B-level for hardware-to-software and software-to-software interfaces are documented in IRSs. The interface requirements documented in one or more HIS or IRS at any level of the specification tree may be organized as appropriate to the system as long as the requirements for a specific interface are not contained in any interface specification at a higher level (including the A-level) and reside in only one HIS or IRS.

3.4.2.1 Prime and Critical Item Development Specifications

The PIDSs contain the requirements for composite prime items (i.e., items decomposable into lower-level composite prime items, HWCIs, CSCIs, or all three) and the methods to be used to ensure that each prime item requirement has been met.

The CIDSs contain the requirements for hardware subsets of prime items that are sufficiently important in their own right to warrant an independent specification and the methods to be used to ensure that each hardware requirement has been met.

---

[22] Although the EIA/IEEE J-STD-016 contents for the IRS does call for some hardware-to-hardware interface information, it is not sufficient to define all types of hardware-to-hardware interface requirements.

3.4.2.2 Hardware-to-Hardware Interface Specifications

The HISs contain the requirements for hardware-to-hardware interfaces between HWCIs and the methods to be used to ensure that each hardware-to-hardware interface requirement has been met.

3.4.2.3 Software Requirements Specifications

The SRSs contain the requirements for CSCIs and the methods to be used to ensure that each software requirement has been met.

3.4.2.4 Software-to-Software and Software-to-Hardware Interface Requirements Specifications

The IRSs contain the hardware-to-software, software-to-software, and software-to-human interface requirements for inter-CSCI and CSCI to HWCI interfaces and the methods to be used to ensure that each interface requirement has been met.

3.5 Design Descriptions

As shown in Figure 13, design descriptions are organized into two distinct sets of documents. The first set is the system architecture description documents. These documents are initiated at the beginning of the system design activity and usually are developed in parallel with the requirements definition activity that results in the requirements specifications. These documents continually are maintained and updated throughout the system development and are critical for maintenance following system delivery. The second set is the design descriptions that are derived directly from B-level specifications and are the basis from which the individual hardware and software components of the system actually will be developed.

3.5.1 System Architecture Design Description

The SSDD and the OCD are complementary documents whose contents must be kept synchronized at all times. The SSDD and the OCD together describe the complete system architecture and how that architecture operates to satisfy the system requirements. The SSDD and the OCD form the conceptual framework used to clarify issues relating to requirements, design, and verification during system development. The system as described in these documents must trace to and satisfy the A- and B-level requirements as documented in the requirements specifications. Lower-level design documents must trace to, and be maintained consistent with, these documents as the design progresses.

After update to as-built documents following system delivery, the SSDD and the OCD form the conceptual framework used by operations and maintenance personnel to understand system operation and to evaluate the impact of all maintenance actions to be performed on the system.[23]

3.5.1.1 System/Segment Design Description

The SSDD describes the system or segment-wide architecture and design for a system or segment (including all computer resources). This description includes documenting the system-wide design decisions that define the system's behavior (i.e., how the system will behave from the user's point of view ignoring internal implementation). This description includes documenting the system architectural design. The architectural design includes identifying the components of the system and showing the static relationships of the components. It also includes documenting the concept of execution among the system components. The concept of execution includes showing how the system components will interact during

[23] See Subsection 4.2 for more information on the use of these documents following system delivery.

system operation from several points of view (e.g., the control, data, sequencing, timing and exception handling points of view). This description also includes documenting the interface characteristics of the system components. The interfaces should include both internal and external interfaces.

3.5.1.2 Operational Concept Description

The OCD documents the system in terms of its relationship to existing systems, that is, its context for use and the ways it will be used in that context. The description of the ways the system will be used include both nominal and off-nominal operation.

This document is used to obtain consensus among the acquirer, developer, support, and user organizations on the operations concept of the proposed system and how the proposed system implements the operations concept contained in the user's external ORD. The OCD describes the existing system and the rationale for change. It describes the new system and its impact on user personnel, support, and operations. It also provides operational scenarios that illustrate the role of the new system, its interaction with users, and its interface to other systems, as well as analysis of the advantages and disadvantages of the new system.

3.5.2 C-level Design Descriptions

The C-level design descriptions are developed by the development organization. As discussed in Subsection 3.3, the C-level hardware and software design descriptions are based on HWCI and CSCI specifications at the lowest level of the specification tree. The C-level hardware and software interface descriptions are based on the interface specification where the interface was documented and may be at any level of the specification tree. The system actually is constructed from the information contained in these documents.

3.5.2.1 Software Design Descriptions

The SDDs describe the design of individual CSCIs that satisfy the requirements contained in the corresponding SRSs and meet the system architecture constraints as documented in the SSDD and OCD. The SDDs describe the CSCI-wide design decisions, the CSCI architectural design, and the detailed design needed to implement the software. Sufficient information on interfaces and database design is presented to ensure a clear understanding of the hardware-to-software and software-to-software interfaces and databases associated with the CSCI design. The specifics of the interface design and database design are documented in the IDD and the database design description (DBDD) documents, respectively.

3.5.2.2 Database Design Descriptions

The DBDDs describe the design of databases. Databases are collections of related data stored in one or more computerized files that can be accessed by users or computer programs via a database management system. The design description may include the software units used to access or manipulate the data. Databases that are used by many design elements generally are documented in DBDDs and may be treated as separate CSCIs. The interfaces to databases may be included in the DBDDs or may be included in separate IDDs.

3.5.2.3 Interface Design Descriptions

The IDDs describe the interface characteristics of one or more hardware-to-software and software-to-software interfaces associated with interfacing hardware and software CIs. The IRS specifies the

interface requirements, while the IDD describes the interface design characteristics selected to meet those requirements.

3.5.2.4 Hardware Product Specifications and Hardware Drawings (Design Descriptions)

Hardware design descriptions have various names and serve different purposes depending on the type of hardware being described, how complicated the hardware is, and how the hardware is to be maintained and upgraded.

Prime Item and Critical Item Product Fabrication Specifications are design descriptions developed from Prime Item and Critical Item Development Specifications, respectively. They are used when both the development and production of a HWCI will be procured and hardware drawings will be required.

Prime Item and Critical Item Product Function Specifications also are design descriptions developed from Prime Item and Critical Item Development Specifications, respectively. They are used when the item to be developed/procured is based primarily on functional (performance) requirements. These specifications (design descriptions) state the complete performance requirements of the item and the necessary interface and interchangeability characteristics. No hardware drawings are developed for these design descriptions. Since the product is described in terms of form, fit and function, any product that meets the descriptions is acceptable.

Hardware drawings relate directly to Prime Item and Critical Item Product Fabrication Specifications and are the blueprints from which the hardware is constructed. Hardware Interface Drawings relate directly to Hardware Interface Specifications and describe the hardware interfaces needed by the hardware to be developed. Hardware Interface Specifications may be included directly in the drawings for each hardware item that uses the interface. When this approach is used, no distinct Hardware Interface Drawings will be developed. Instead, the Hardware Interface Specifications will relate directly to each Product Fabrication Specification or Product Function Specification that uses the interface.

3.5.2.5 System/Segment Interface Design Descriptions

The SSISs should be structured to segregate hardware-to-hardware interfaces from hardware-to-software, software-to-software, and system-to-human interfaces. This allows the system/segment interface designs to be decomposed into associated Hardware Interface Drawings for the hardware-to-hardware interfaces and IDDs for the hardware-to-software, software-to-software, and system-to-human interfaces.

3.6 System Verification Documentation Introduction

Each acquisition requires an acquisition-specific hierarchy of verification products that relates to the hierarchy of specification and design products discussed beginning in Subsection 3.2. This verification product hierarchy is planned and controlled at the system level by the System Integration and Verification Master Plan discussed in Subsection 2.5.2.1, at the software level by the Software Master Build Plan discussed in Subsection 2.6.1.1, and at the hardware level by the Hardware Integration and Verification Master Plan discussed in Subsection 2.6.3.1.

As discussed in Subsection 3.4, each specification contains a qualification section that defines at what level or levels of system integration each requirement will be verified and the verification method[24] or methods to be used for each requirement at each level.

---

[24] A verification method can be one of four types: Inspection, Analysis, Demonstration, or Test. Definitions for each of these methods are included in Appendix A of this report.

In order to describe the content and relationships among the verification product types, a representative verification product hierarchy has been defined in Subsection 3.7. This representative verification tree contains one of each type of verification product normally used by SMC/NRO acquisitions. In order for these products to be used for a specific acquisition, the relationships and content described here must be translated appropriately into that acquisition's specification tree and integration and verification approach.

Verification plans define how the qualification of requirements will be implemented. Verification descriptions define how the verification methods associated with each requirement in the specification will be performed. Verification reports document the results of the verification process. The Master Verification Record contains the cumulative record of the verification status of all requirements in each specification in the specification tree. The Master Verification Record is a component of the Traceability/Accountability Repository discussed in Subsection 3.9.

3.7 Representative Verification Documentation Tree

In the interest of simplicity, the representative system for the verification documentation tree will be constructed from a two-level integration structure and a three-level specification hierarchy. Since this report has assumed that the same verification document structure of plans, descriptions, and reports applies to all levels of verification, this representative verification documentation tree can be generalized to any integration structure and specification hierarchy independently of whether the elements are hardware, software, or compound hardware and software elements.

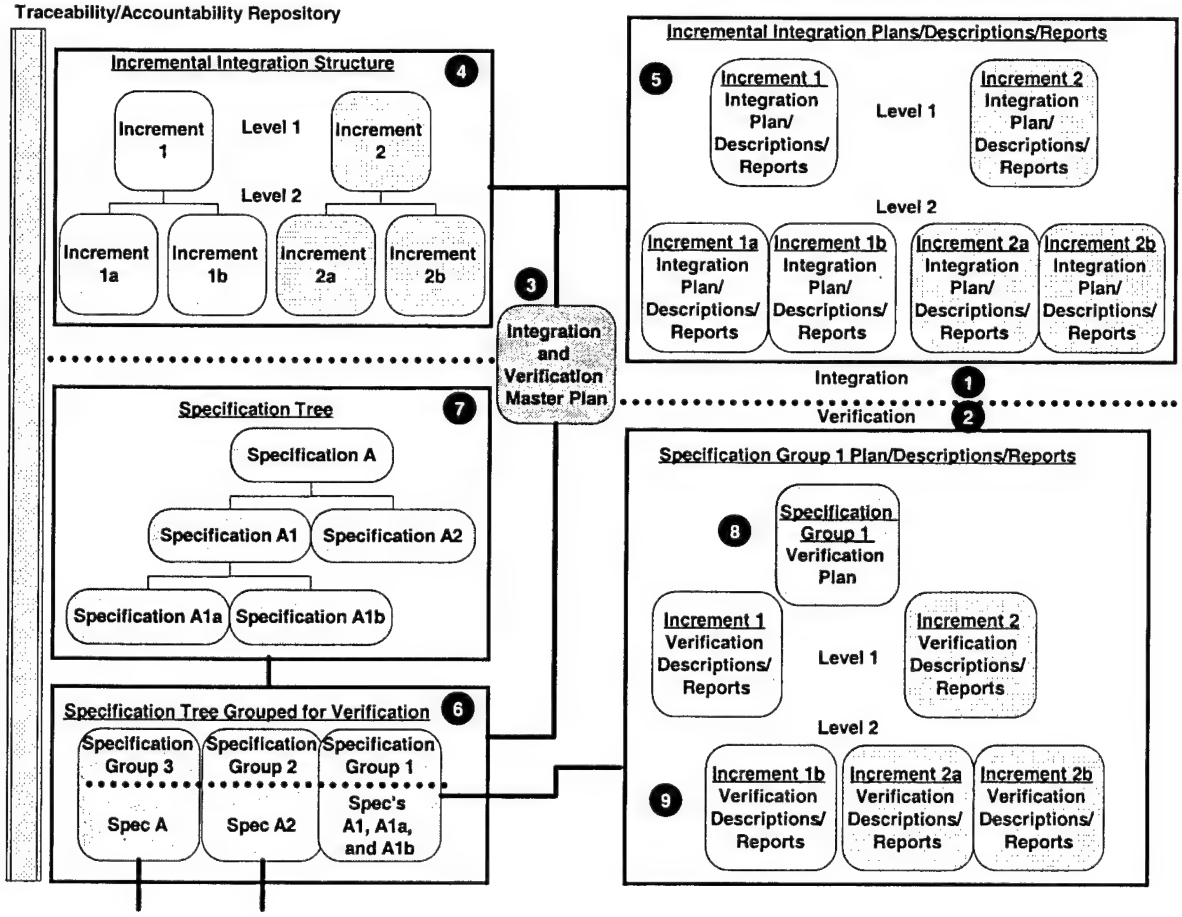


Figure 17. Integration/specification/verification relationship.

The following discussion references Figure 17. Where appropriate to the discussion, associations between the text and different portions of the figure will be made using corresponding numbers in each.

The figure is divided into two halves. The upper half is identified as "Integration," ❶ and the lower half is identified as "Verification."❷ How the incremental integration structure and the incremental integration plans, descriptions, and reports are developed for the integration half of the figure has been discussed in Subsections 2.5.2.1, 2.6.1.1, and 2.6.3.1. In this representative system, the planning processes and procedures have resulted in the development of an Integration and Verification Master Plan ❸ that defines the relationships among the Incremental Integration Structure ❹, the Incremental Integration Plans, Descriptions, and Reports ❺, and the Specification Tree Grouped for Verification ❻. The Incremental Integration Structure ❹ consists of two levels. The first level comprises two increments, and the second level comprises four increments. For each increment at each level, there are associated incremental integration plans, descriptions, and reports ❺.

Successfully implementing a system that meets the requirements contained in the specifications in the specification tree ❼, successfully performing the integration of the defined increments ❹ using the appropriate plans and descriptions, and documenting the results in the appropriate reports ❺ will result in a system that performs acceptably for operational use. However, in order to provide objective evidence to independent observers that the system requirements actually have been met, a series of formal verifications must be performed independently of the integration effort.

The planning in the Integration and Verification Master Plan ❸ includes the assignment of complete specifications in the specification tree to specification groups ❻. In the representative system, the specifications A1, A1a, and A1b are assigned to Specification Group 1. The specification A2 is assigned to Specification Group 2 and specification A is assigned to Specification Group 3. The assignment of specifications to groups may come from many factors (e.g., Specification Group 1 may be a subcontractor, Specification Group 2 may be a peer to the prime contractor, and Specification Group 3 may be the prime contractor), but however they are determined, the assignment must result in all specifications in the specification tree being assigned to some specification group, and each specification must be assigned in its entirety to only one group.

Each specification group then can have its verifications planned ❽. Over time, this planning results in the development of verification descriptions that will be performed following the successful integration of the corresponding integration increment ❺. The results obtained from the verification descriptions' performance are documented in verification reports that describe whether or not the requirements that were intended to be verified indeed were objectively verified. Not every integration increment needs to have a corresponding independent verification. In the planning for the representative system, it has been determined that verification will not be performed on Integration Increment 1a of Level 2 ❾.

The following discussion references Figure 18. Where appropriate to the discussion, associations between the text and different portions of the figure will be made using corresponding numbers in each.

Figure 18 is a more detailed explanation of the relationship of Specification Group 1's specifications, its verification plan, and its associated verification descriptions and reports that were shown in Figure 17 ❻, ❽, and ❾.

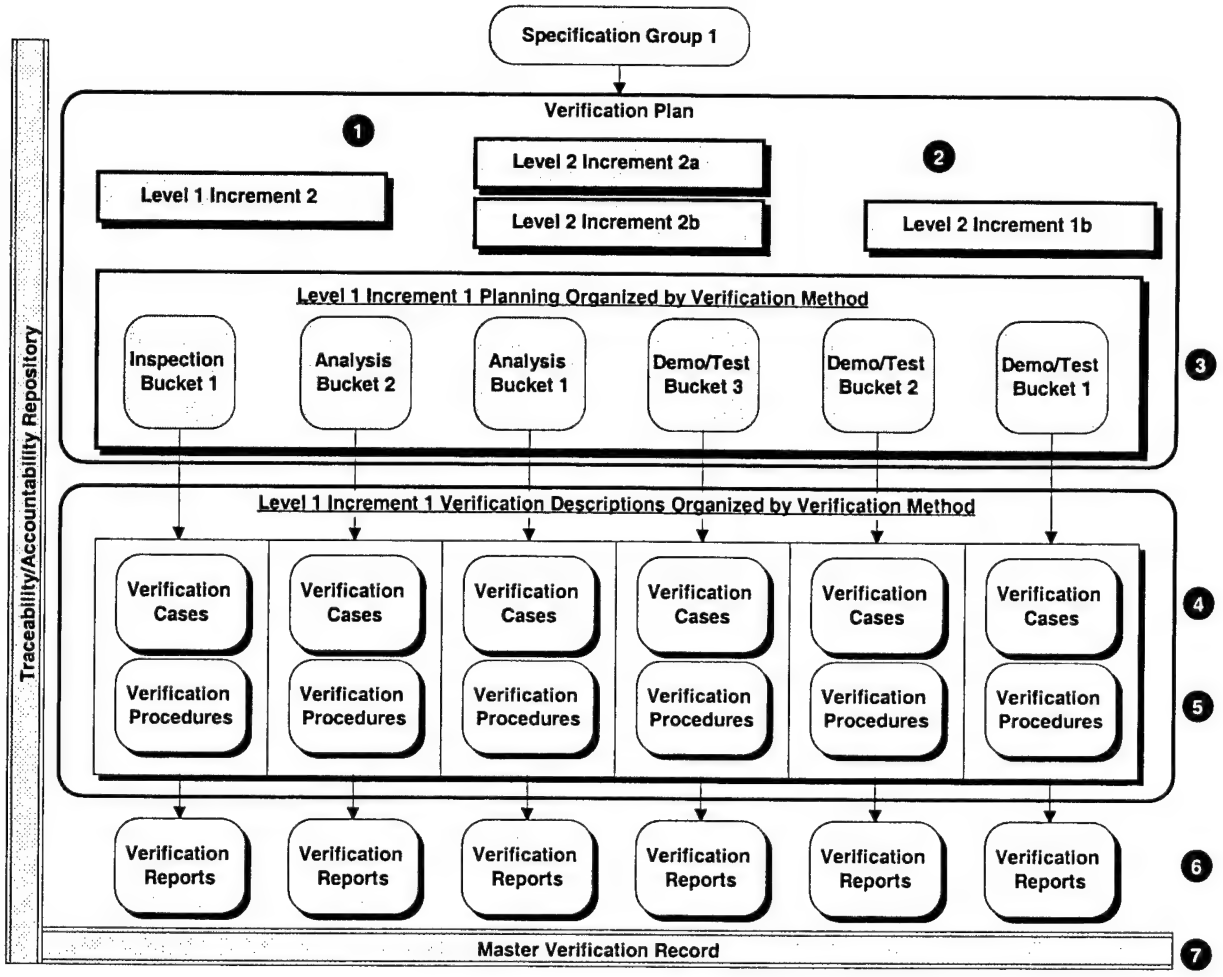


Figure 18. Specification/verification relationship.

Each specification assigned to Specification Group 1 (Specifications A1, A1a, and A1b) contains a set of requirements. At the end of each specification is a qualification section that relates each requirement in the specification to a verification level or levels. The verification level corresponds to a specific integration level and increment. The qualification section also indicates the verification method or methods to be used to verify the requirement at each verification level.

The verification plan encompasses the verification planning for all of the requirements within Specification Group 1 at whatever integration level and increment the requirements are to be verified. This approach ensures that all requirements in meaningfully related groups of specifications are planned as a whole in one place across all verification levels and for all applicable verification methods.

This comprehensive planning is particularly critical when the characteristics of some requirements dictate that they must be verified in multiple verification levels (i.e., in multiple integration levels and increments) using multiple verification methods in order to determine completely that the requirements actually are met. In this case, the verification of the requirements at a single verification level using a specific verification method will only contribute to determining that the requirement actually is met. Until all verification levels and all verification methods have been performed successfully, the status of the requirement must be considered only partially or conditionally verified.

To perform verification planning, the requirements in the Specification Group 1's specifications first are organized by verification level so that each requirement in each specification is associated with one or

more specific integration levels and Increments ❶. In the representative system, the specifications in Specification Group 1 have requirements that are to be verified at each integration level and increment except for Level 2, Integration Increment 1a ❷. Once the requirements in the specifications have been organized by integration level and increment, the verification planning for each can be performed.

Within each integration level and increment group are requirements that are assigned one or more of four verification methods (i.e., Inspection, Analysis, Demonstration, and Test[25]). Verification planning at this point consists of (a) grouping the requirements associated with each verification method together and (b) determining how many subdivisions (buckets) should exist for each verification method. In the representative system, the requirements to be verified following the Level 1, Increment 1 integration are subdivided into six buckets: three groups for Demonstration and Test, two groups for Analysis, and one group for Inspection ❸. The division of the requirements associated with a verification method into multiple buckets depends on the specifics of a particular system. In some cases, multiple buckets are created because natural functional groupings exist that make the verification more efficient. In some cases, multiple buckets are created because it makes sense to perform these groups at different locations or at different times.

Once the requirements in a specification are associated with a specific bucket, each bucket then can be planned. The planning within a specific bucket consists of a description of the verification environment to be used. For example, the site for the verification, the configuration of any material needed to support the verification (i.e., documentation, hardware, and software), the verification participants, and the specific verification events to be performed. Planning for the verification events to be performed includes defining the number, organization, scheduling, unique identification, and data collection for each verification event. Performing the set of planned verification events will show that the requirements associated with this bucket have been satisfied. For example, the verification environment for a specific demonstration/test bucket might consist of a specific configuration of a specific integration testbed at a specific site using a specific set of test cases. The verification environment for a specific analysis bucket might consist of a specific configuration of a specific simulation system at a specific site using a set of simulation runs.

Once the verification events have been determined, the verification descriptions for each event can be developed. Verification descriptions generally are performed later in the development life cycle and consist of two distinct sections.

The first section is called the Verification Cases ❹. This section relates directly to the verification events documented in the verification plan. This section expands the information about each verification event into its specific verification preparations and characteristics. The verification preparations for each event include hardware and software preparation and document preparation as applicable to the method. The verification characteristics for each event include a list of the requirements to be addressed, the required prerequisites, the required inputs and expected outputs, and the criteria for evaluating the results. This section generally is developed prior to the completion of integration.

The second section is called the Verification Procedures ❺. This section relates directly to the verification cases described in the first section. This section defines the specific steps that must be followed in order to perform the verification case defined in the first section. The information contained in this section is documented in sufficient detail that the exact conditions of the verification can be recreated and the verification steps can be repeated accurately. This section generally is developed just prior to performing the verification.

---

[25] Definitions for each of these methods are included in Appendix A of this report. For the purposes of this report, the Demonstration and Test verification methods will be combined in the verification discussion, since the class of information needed by each essentially is identical.

Once the verification procedures for each verification case have been performed, a report of the results is developed ❻. This report provides a comprehensive overview of the verification results as a whole and the detailed results for each verification case. The detailed results for each verification case include a summary of the event, a list of problems encountered and their impact on the validity of the event, and a record of the event activities. This report also identifies the status of the requirements intended to be verified by the event. For example, a verification event that encountered no problems would produce a report that indicated that the status of the requirements in the event is verified. A verification event that encountered problems where the verification of some of the requirements could not be determined would produce a report that indicated that some of the requirements were verified and some of the requirements were unverified.

The meaning of the verification status of a requirement as indicated in any single report must be understood in the context of that requirement's total verification needs. For example, the verification status of a requirement that needs only to be verified in one verification level using one verification method will be determined completely by the report for that level and method. However, the verification status of a requirement that needs to be verified in more than one verification level and/or using more than one verification method will be only partially determined by the report for one level and method. Since it is generally the case that requirements are verified at more than one level and/or using more than one method and that different levels and methods are performed at different times and locations by different organizations, it is critical to the successful conclusion of any verification program that a Master Verification Record be maintained that accumulates the status of requirements as successive verification levels and methods are completed ❼. It also is critical that traceability and accountability be maintained between the status of a requirement and the supporting evidence for that status. The Master Verification Record is one component of the Traceability/Accountability Repository.

The Traceability/Accountability Repository is indicated by a symbol at the left in Figures 17 and 18. This symbol represents one or more information structures that contain the system-wide repository for certain critical information and the bi-directional traceability of certain critical information developed over the life of the system. Although the complete definition of what information needs to be traceable and accountable will be unique to a specific program, a certain minimum set must be included. For the purposes of this report, that minimum set consists of traceability and accountability related to the documentation set contained in Figure 13. The characteristics of the minimum set of traceability and accountability required for any program are described in Subsection 3.9.

3.8 Verification

In general, the verification document tree is developed as a function of the structure of the organizational groups, the structure of the integration levels and increments, and the system specification tree. The organizational groups define the major divisions of responsibility for system development as well as the primary points of transition between responsible parties. The integration levels and increments define the primary components from which the system will be constructed. The specification tree defines the requirements that the system must meet in order to be accepted. It also defines at what level or levels in the system integration process and by what method or methods requirements will be verified.

Analyzed as a whole, these factors result in the organization of the specification tree into unique specification groups. Verification of the requirements in these specification groups is planned as a whole. The planned verifications take place at specified points in the integration process where objective, independent determination is made that requirements are being met.

The verification documentation tree therefore is the same at any point in the verification process. Planning for the verification of requirements in a specification group is documented in the Verification Plan; the process and procedures to implement the planned verifications are documented in the

Verification Descriptions, and the results from execution of the verification procedures are documented in Verification Reports.

3.8.1 Verification Plan

The verification plan defines how the verification of the requirements in the specification group will be structured and organized. It encompasses the verification planning for all of the requirements in the specification group. This planning covers the entire range of verification levels and verification methods defined for the requirements in the specification group. This planning also determines the environments for the verification such as the sites to be used; the hardware, software, and documentation to be used; the participating organizations; and the specific verification events to be performed. Planning for the verification events includes defining the number, organization, scheduling, unique identification, and data collection needed for each event. Successfully performing the planned events will result in the verification of each requirement in the specification group.

The verification plans must be consistent with the associated implementation and integration plans for both the system elements to be verified by the plan and the specific tools, environments, and data/scenarios required to support the verification.

3.8.2 Verification Descriptions

Once the set of verification events has been determined by the verification plan, the verification descriptions for each event can be developed. Verification descriptions are developed in two independent sections that are separated in time.

The first section is called the Verification Cases. This section relates directly to the verification events documented in the verification plan. It expands the information about each verification event into its specific verification preparations and characteristics. The preparations for each event include hardware and software preparation and document preparation as applicable to the verification method. The verification characteristics for each event include a list of the requirements to be addressed, the required prerequisites, the required inputs and expected outputs, and the criteria for evaluating the results. This section generally is developed prior to completing integration.

The second section is called the Verification Procedures. This section relates directly to the verification cases described in the first section. This section defines the specific steps that must be followed in order to perform the verification case defined in the first section. The information contained in this section is documented in sufficient detail that the exact conditions of the verification can be recreated and the verification steps can be repeated accurately. This section generally is developed just prior to performing the verification.

3.8.3 Verification Reports

Once the verification procedures for each verification case have been performed, a report of the results is developed. This report provides a comprehensive overview of the verification results as a whole and the detailed results for each verification case. The detailed results include a summary of the event, a list of problems encountered and their impact on the validity of the event, and a record of the event activities.

This report also identifies the status of the requirements intended to be verified by the event. Specifically, the status of a requirement following the execution of verification procedures is either verified or not verified, depending on the success or failure of the verification procedure. However, when a specific requirement must be verified at more than one level or by more than one method, the meaning of the status derived from any one report must be considered from a more comprehensive perspective. Until a

requirement has been verified successfully at each applicable level and by each applicable method, that requirement must be considered only partially (or conditionally) verified. The comprehensive status of the requirement must be maintained in the Master Verification Record as each verification event that affects the requirement occurs. The Master Verification Record is a component of the Traceability/Accountability Repository. The Traceability/Accountability Repository is discussed below.

3.9 Traceability/Accountability Repository

The Traceability/Accountability Repository is a collection of information structures that provide traceability relationships among information critical to the success of any system development. The repository also provides accountability for the allocation, status, and rationale for essential quantitative and qualitative information. The contents of these information structures may reside in one or more physical repositories at one or more sites. However, it is absolutely essential that the information contained in these information structures be correct and consistent for effective program management to occur. This subsection will provide a description of the information that must be maintained in the repository.

Although the complete definition of what information needs to be traceable and accountable, and how the information physically is structured and located is unique to a specific program, there is a certain minimum set of information that must be included in the repository. For the purposes of this report, the minimum set of traceability and accountability information is the traceability and accountability related to the set of documents depicted in Figure 13. These documents include requirements specifications, interface specifications, system architecture and operations descriptions, design descriptions, integration planning, and verification documentation.

It is essential for the successful execution of any system development that unambiguous traceability and accountability of requirements sources, architectural decisions, implementation designs, quantitative allocations, integration increment content, and verification dispositions be accurately developed and maintained throughout the life of the system. Without this accountability, no objective assessment can be performed about the correctness, completeness, or adequacy of the system with respect to requirements, design, implementation, integration, or verification either during development or at acceptance.

The Traceability/Accountability Repository is discussed separately from specific system products in recognition of the critical role that unambiguous traceability and accountability plays in system development and maintenance. When products in other sections of this report discuss traceability, accountability, or relationships between information, it is assumed that either the document references the appropriate information in the Traceability/Accountability Repository or the appropriate information is extracted from the Traceability/Accountability Repository and inserted into the document so that all traceability/accountability information is maintained centrally rather than in each document. This approach ensures a consistent and complete traceability/accountability of this critical information over the life of the system.

For the purpose of this report, the term traceability is used when the bi-directional association of one item with another is used primarily to establish and maintain the relationships between the items. These relationships can be either hierarchical or peer-to-peer, depending on the purpose of the relationship being established. The term accountability is used when the bi-directional associations of items with each other are used to establish both the relationships between the items and to associate an item with its current status or quantitative value. The bi-directional associations used for accountability also may be either hierarchical or peer-to-peer.

The following discussion references Figure 19. Where appropriate to the discussion, associations between the text and different portions of the figure will be made using corresponding numbers in each.

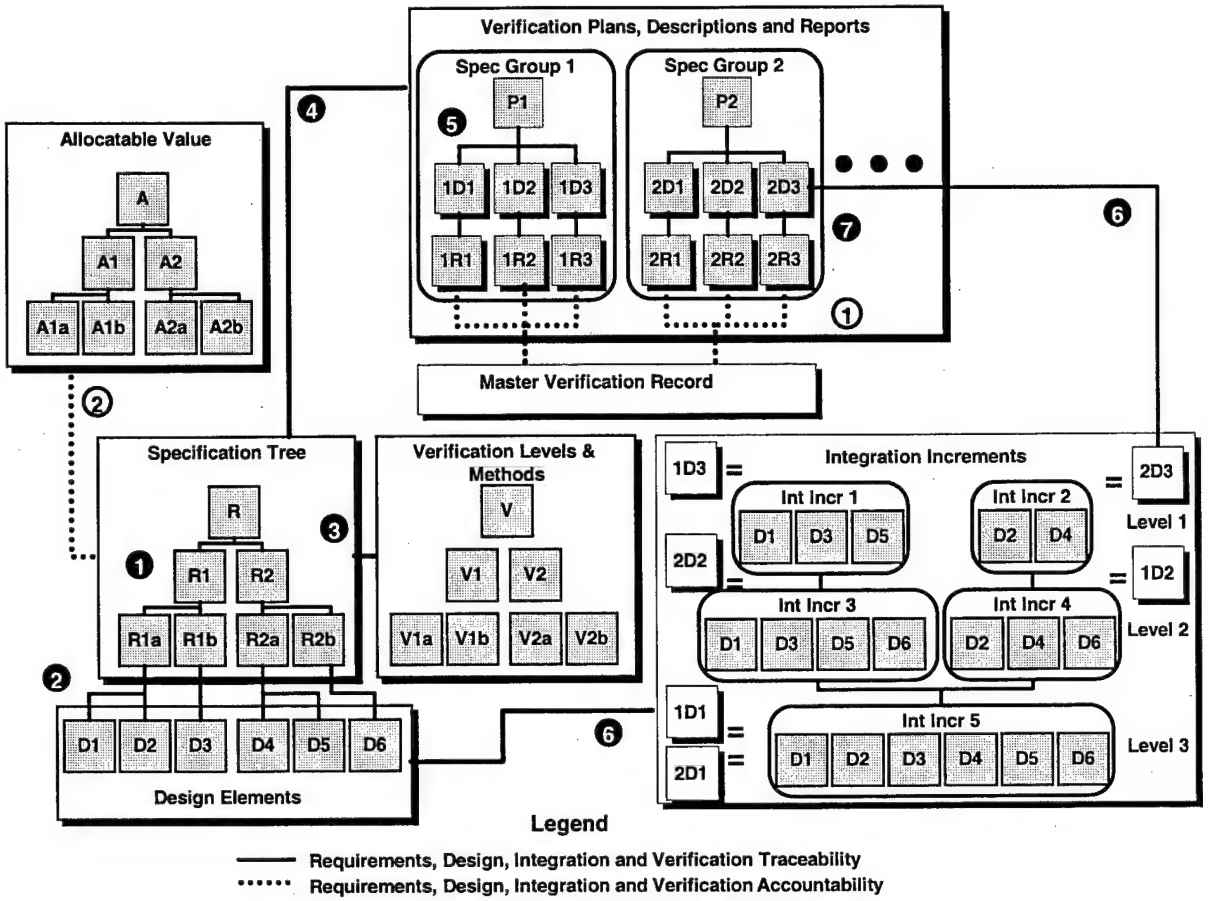


Figure 19. Requirements, design, integration, and verification traceability.

The information contained in the Traceability/Accountability Repository is necessary for both development and maintenance. This information consists of bi-directional traceability and accountability for the following types of information:

a. **Requirement-to-requirement traceability ❶:** This traceability ensures that all requirements to be met by the system are derived from some authorizing source. Requirement-to-requirement traceability includes the case in which a requirement is derived from some authorized design decision. In this case, the design decision acts as an authorized source.

b. **Requirement-to-design traceability ❷:** This traceability ensures that all design elements used to meet system requirements are derived from some authorizing source.

c. **Requirement-to-verification level and method traceability ❸:** This traceability ensures that each requirement in each specification has at least one assigned place in the system integration structure where it will be verified and at least one verification method that will be used to perform the verification.

d. **Requirement-to-verification plan traceability ❹:** The requirements in the specifications, Rx, of the specification tree are distributed to the verification plans, Py, of specification groups. This traceability ensures that each requirement in each specification is included in one and only one verification plan.

e. **Requirement-to-verification descriptions traceability ❺:** This traceability consists of two parts. In the first part, this traceability ensures that each requirement allocated to a verification plan is included in all appropriate verification cases contained in the verification descriptions. The appropriate verification cases in the descriptions are those that encompass the verification levels and verification methods associated with that requirement in its specification. In the second part, this traceability ensures that each requirement in a verification case is traceable to a specific set of verification procedure steps that, when successfully performed, will result in the verification of that requirement from the point of view of the specific verification case.

f. **Requirement/verification level-to-integration increment traceability ❻:** Requirements to be verified are associated with specific integration increments through verification descriptions. The verification descriptions, e.g., 2D3, are associated with specific integration increments, e.g., Int Incr 2, that contain specific design elements, e.g., D2 and D4. This traceability ensures that the capabilities of each integration increment are adequate to verify the requirements contained in the associated verification descriptions.

g. **Requirement-to-verification report traceability ❼:** This traceability ensures that the results of performing the verification steps contained in the verification description are recorded for each requirement included in the verification description.

h. **Requirement-to-verification status accountability ①:** This accountability, called The Master Verification Record, ensures that the current verification status of each requirement accurately reflects the cumulative status resulting from performing the verification steps in the descriptions that contain that requirement.

i. **Quantitative requirement accountability ②:** This accountability ensures that all quantitative requirements (e.g., reliability, availability, operational timelines) are allocated to the appropriate specifications and that the value allocated to each specification is appropriate. Since a quantitative requirement is just one specific type of requirement, it has the same traceability associated with it as all other requirements discussed previously (e.g., requirement-to-design, requirement-to-verification level and method, requirement-to-verification plan).

j. **Interface association traceability:** This traceability ensures that the requirements for each interface are associated with the requirements of all other related interfaces. This association ensures that the relationships among all interfaces are known and that changes to any specific interface can be analyzed accurately for impacts to all related interfaces. Interface association traceability is sufficiently complex that an expanded explanation has been included in Appendix B.

As development proceeds on a program, changes occur that affect cost and schedule. These changes may be caused externally or internally, but each must be evaluated for its effect on the program. The traceability and accountability described in this section provide the basic framework for the analysis of the total program impact of these changes. For example, if requirements changes have occurred and the traceability and accountability information is accurate, then the changes can be traced to the affected design, integration increments, and verification plans, and the cost and schedule impacts of the changes can be determined effectively.

4. System Operations and Maintenance Relationships

This section describes the relationships between system operations and maintenance products. The text in subsequent paragraphs refers specifically to elements in Figure 20.

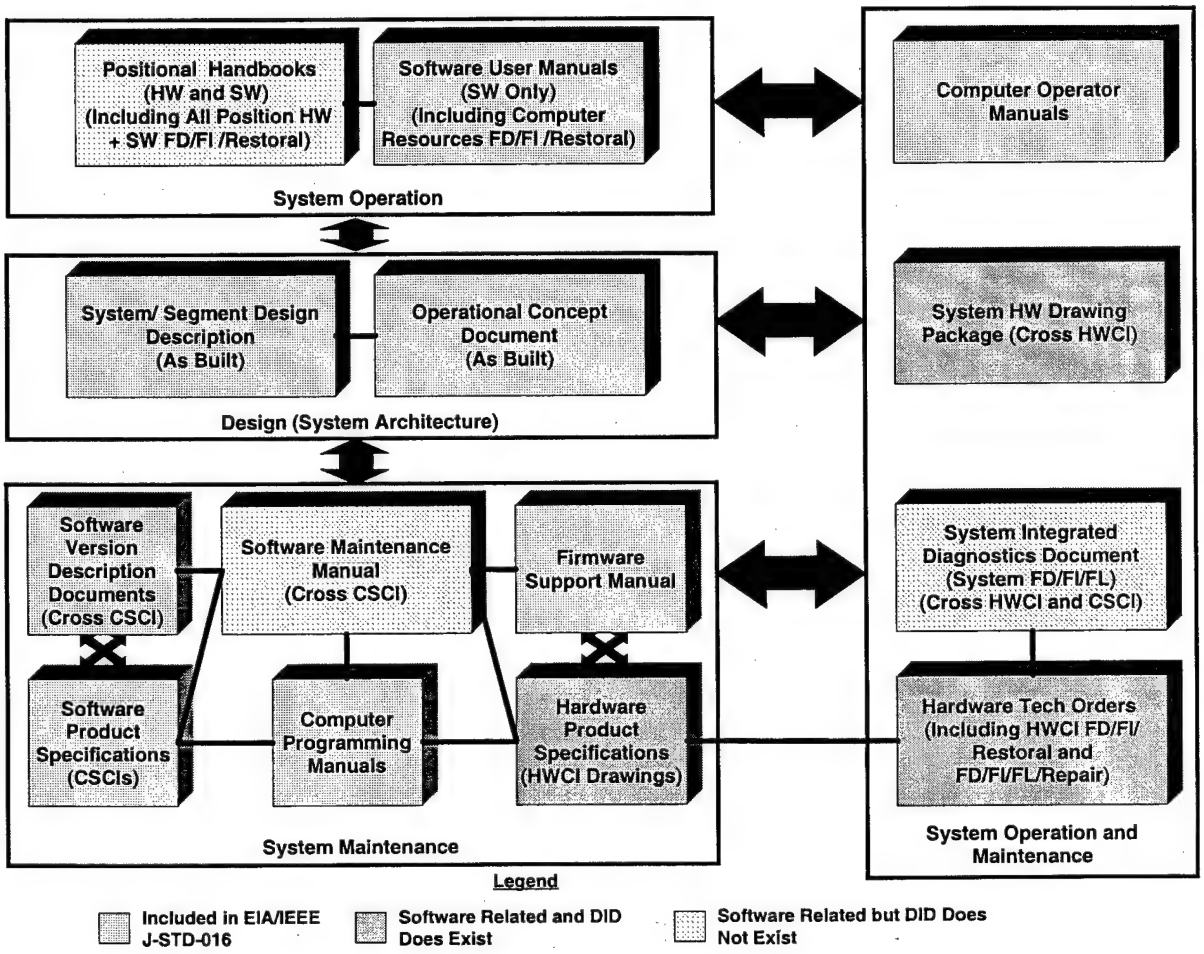


Figure 20. System operations and maintenance relationships.

The "System Operation" box includes all of the operator positions that directly or indirectly support the primary purpose of the system. Examples of operational positions include operators who perform specific functions related to the system payloads (for satellite systems), operators who perform communication setup and monitoring (for remote sites), and operators who control and monitor networks internal to ground stations (for distributed processing). These operators perform nominal operations related to the scope of their responsibilities in the system as well as certain administrative operations and non-nominal system maintenance (System Restoral) operations. An example of an administrative operation might include the operators performing software or database updates at the individual consoles/workstations within the scope of their responsibilities. System operator non-nominal maintenance operations are discussed in the next paragraph. The "Design (System Architecture)" box contains the as-built maintenance versions of the documentation shown in the "System Architecture" sub-box in the "Design" box of Figure 13.

The System Maintenance box contains the information needed for performing computer resources maintenance. This maintenance is divided into two very distinct functions. The first is maintenance in the sense of performing Fault Detection (FD) and Fault Isolation (FI) in order to restore the computer resources to some operational state (even if it is in some form of degraded mode) as quickly as possible. This form of maintenance is called "System Restoral" and is performed by both operations and maintenance personnel. The term System Restoral applies to all system functions, both hardware and software. The second is maintenance in the sense of performing FD, FI, and Fault Localization (FL) in order to repair a fault in the computer resources. This form of maintenance is called "System Repair" and is performed by maintenance personnel. The term System Repair applies to all system functions, both hardware and software. The set of actions required to perform the two types of maintenance functions will be referred to as system restoral and system repair. The "Systems Operation and Maintenance" box contains two types of information. The first is additional information needed to support nominal operations for both computer resources and non-computer resources. The second is the complementary system restoral and system repair information needed for the non-computer resources portions of the system.

As can be seen from the figure, some system products support nominal operations and both the system restoral and system repair maintenance functions, while others relate to either operations or maintenance. As shown in the figure, the software-related products are not self-contained within the total set of system products. This is an artifact of the original structure of the software-centered philosophy embodied in MIL-STD-498 and its predecessor standards. The implication of this is that considerable additional work at the system level is required to ensure a consistent set of products for a specific program.

Three products that are very critical to the successful operation and maintenance of the system do not have existing DIDs that could be used as the basis for tailoring the system products. These products are identified here as the Positional Handbook, the System Integrated Diagnostics Document, and the Software Maintenance Manual.

4.1 System Operations

4.1.1 Positional Handbook

A Positional Handbook treats a particular position from a system perspective and defines and describes nominal, administrative, and non-nominal tasks for both hardware and software that are within the scope of responsibility of that specific operator position. The existing software user manual discussed below provides information only for computer resource-related responsibilities.

4.1.2 Software User Manual

A Software User Manual (SUM) serves two primary functions. It can be organized to provide a hands-on software user with the information needed on how to install and use a CSCI, a group of related CSCIs, or a software system or subsystem.

It also can be organized to treat a particular position from a computer resources perspective and describes nominal, administrative, and non-nominal tasks for the computer resources (computer-related hardware and software) that are within the scope of responsibility of that specific operator position. When the SUM is organized to support a specific operator position, other non-computer-related hardware associated with the position is not covered. Since many operator positions entail both computer resource-related and non-computer resource-related responsibilities, it generally is preferable to use a Positional Handbook to describe the total scope of position responsibilities rather than have two separate documents.

4.2 Design (System Architecture)

Following system delivery, the "as built" SSDD and OCD form the conceptual framework used by operations and maintenance personnel to understand system operation and to evaluate the impact of all maintenance actions to be performed on the system (corrective maintenance, enhancement maintenance, and adaptive maintenance).

4.2.1 System/Segment Design Document

The "as built" SSDD describes the system-wide architecture and design for the delivered system. It contains the "as built" information described in Subsection 3.5.1.1.

4.2.2 Operational Concept Document

The "as built" OCD describes the system in terms of its relationship to existing systems and the ways it will be used in those operations and maintenance environments. It contains the "as built" information described in Subsection 3.5.1.2.

4.3 System Maintenance

4.3.1 Software Maintenance Manual

The Software Maintenance Manual contains the information needed to maintain the software in the system as an integrated whole. The term "software" used here is intended in its broadest sense. As such, it includes all code written in any language, all code defining the structure of databases, all code/instructions used by any COTS software package, and all data necessary for the execution of the software whose modification/update is not covered in other documentation such as the Positional Handbook or Software User Manual. The data included here might consist of the contents of files or database tables that are not modifiable by the user. These tables might be used for such things as adaptation/configuration of the system for different locations or environments, or system tuning.

In general, the software in a specific system has natural groupings above the CSCI level that are meaningful from the point of view of software maintenance. These groupings may contain software that requires multiple maintenance environments. For example, all ground software may be maintained as a whole from one location. That software may require a database maintenance environment and a firmware maintenance environment as well as a Higher Order Language maintenance environment. Each of the logical groupings of software that comprise a system and each of the maintenance environments required by those logical groupings needs to be included in the Software Maintenance Manual.

When there are multiple logical groupings of software or software maintenance environments, then a primary manual provides an overview of the total set of logical software groupings and associated maintenance environments and their relationships to each other. Each of the logical software groupings and specific maintenance environments then may be described as appendices within the primary document or as separate documents referenced from the primary document, as appropriate to the system to be maintained. The overview of the hardware and software architecture provided in the primary document must be sufficient to understand the subsequent material in that document, its appendices, and referenced documents. Information already present in other documents such as the SSDD should be referenced rather than repeated.

Each appendix or separately referenced document in the Software Maintenance Manual[26] describing a logical grouping of software should contain the following topics:

a. **Hardware and Software Architecture Overview:** The information in this section describes the hardware and software architecture of the system to the extent required to understand subsequent material for this logical grouping of software.

b. **Software Maintenance Environment Description:** The information in this section complements (but does not duplicate) the primary document and the SSDD by documenting the software engineering environment computer resources required to provide software maintenance (e.g., software repair) for the software in this logical grouping. This information includes a description of the hardware configuration, the operational and support software available, and the relationship between the hardware and software. This information also includes any test hardware and software (e.g., simulators) or interfaces to external systems (e.g., communication links).

c. **Modification Procedures:** The information in this section describes how to modify the software in this logical grouping for every unique environment in which the software must execute (e.g., multiple operational sites and multiple maintenance environments). This includes the operational software, operational support software, and any software necessary for actual software maintenance. The information in this section should not duplicate the information contained in Software Development Folders or equivalent documents. The information in this section includes or references:

   (1) The requirements analysis, design, coding, and verification standards and conventions used

   (2) The transformation procedures (such as compilation and build procedures) used for this logical grouping of software that are not included in the SPS as part of the product but are needed for maintenance

   (3) The integration and verification procedures to be followed and associated workloads to be used including any standard regression test and/or verification suites

   (4) Support facilities, equipment, software (both COTS and developed, whether deliverable or non-deliverable) and the transformation procedures (such as compilation and build procedures) necessary to modify that software

   (5) Databases or other data files used by the software, their contents, and procedures for using them in the context of the maintenance environment

   (6) The traceability between the above information and specific adaptations of the software products as documented in the SPSs for the system

d. **Installation Media Creation Procedures:** The information in this section describes how to create the media to be used for installing the software products for this logical grouping of software in each unique environment in which the software must execute (e.g., operational, operational support, and software maintenance environments). This includes any variations necessary for different sites. The installation media created using these procedures is used as input for the installation instructions contained in the Software Version Description document and the Firmware Support Manual.

e. **Software Fault Detection/Fault Localization/Fault Isolation Procedures for Repair:** The information in this section complements (but does not duplicate) the System Integrated Diagnostics

[26] As an alternative to a separate Software Maintenance Manual or equivalent, the topics included in the "as built" Software Maintenance Manual could be distributed as follows: (a) The Hardware and Software Architecture Overview and the Software Maintenance Environment Description should be included in the SSDD, (b) The Modification Procedures, the Installation Media Creation Procedures, and the Software Fault Detection/Fault Localization/Fault Isolation Procedures for Repair should be included in the appropriate Software Product Specification.

Document by documenting the cross-CSCI and CSCI-level procedures needed to detect, localize, and isolate software faults in the operational software, the operational support software, and in any software needed to support maintenance. The information in this section also does not duplicate fault detection, fault localization, and system restoral information contained in Positional Handbooks or Software User Manuals.

4.3.2 Software Product Specifications

The development of modern software systems has created complications in the identification and control of software products both during development and after delivery to the government. The complication arises in the use of modern software architectures, such as distributed systems, open systems, and layered architectures–and methodologies such as object-oriented analysis and development methodologies.

Under the current philosophy for the identification of a software product as used in EIA/IEEE J-STD-016 and its associated DIDs, the atomic unit for the software product is the CSCI. The deliverable software product under this philosophy consists of the source-level elements[27] of the CSCI, specific transformation procedures[28] (such as compilation/build procedures), and the resulting executable(s). This philosophy assumes that only source elements within a CSCI are combined by the transformation procedures (such as compilation/build procedures) to create the CSCI's executable(s). In other words, any executable created from a CSCI does not require elements of any other CSCI. The current philosophy does provide for the existence of cross-CSCI interactions in the transformation procedures (such as the compilation/build procedures) as contained in the SPS DID but does not recognize that the resulting executable(s) may not be allocable directly to any one set of CSCI source-level elements.

SPS tailoring developed for a specific program should recognize the impact of new software architectures and methodologies on the identification and description of the software products produced and should be structured to allow the software product description contained in the SPS to accurately reflect that impact. This report assumes that the software product descriptions will be organized around the smallest set of items (source-level elements, transformation procedures, and executables) that are self-contained from the point of view of being used or using any item outside the set, as shown in Figure 21. In other words, if a given executable is created from multiple CSCIs, then all of those CSCIs and the associated transformation that create that executable are a part of the software product and will be included in a single SPS. Furthermore, if the CSCIs that created the executable also participate in the creation of other executables, then the other executables, the associated transformation procedures, and any new CSCIs that participate in the creation of the other executables also will be included in that same SPS.

---

[27] Source-level elements associated with a CSCI in this discussion include all source language, data not intended to be user-populated and modified, database structure descriptions, expert system rules, COTS products, et cetera, needed by the CSCI to allow the appropriate transformation procedures to create the associated executable(s). They also include all data needed by the executable(s) in order to perform their intended function unless those data are transient in nature, included in some other CSCI, or the data are intended to be user populated and modified. The decision to include data as part of a CSCI versus not in any CSCI generally is a design decision that is system specific and documented in the SSDD or the SDD. For example, the data needed to populate a vehicle specific database probably would not be included in any CSCI, while the data needed to tune the database performance would be included in some CSCI.

[28] Transformation procedures associated with the conversion of source-level elements into executable elements depend on the type of source-level element under consideration. For example, Ada or C++ source language elements are transformed into executables using compilation/build procedures appropriate to the development environment used. In some cases where 4GL languages are used, the transformation procedures may include several steps such as the preprocessing of the 4GL source into C++ source followed by compilation/build procedures in order to create an executable. In some cases, the transformation procedures are the identity transformation, i.e., the source-level element and the executable-level element are identical. This generally is the case for certain types of data that are not physically transformed to participate in an executable.

This process will create a set of disjoint groupings of CSCIs, their associated transformation procedures, and the executables created by those transformation procedures. Each of these disjoint sets will have an SPS. An item in any of the disjoint sets is called a product element.

The only time an SPS will contain a single CSCI with its corresponding set of transformation procedures and executable(s) will be when that single CSCI does not require any source elements from any other CSCI to participate in the creation of its corresponding executable(s) and when no source element from the original CSCI is used by any other CSCI's transformation procedures for the creation of its executable(s).

The following discussion references Figure 21. Where appropriate to the discussion, associations between the text and different portions of the figure will be made using corresponding numbers in each.

In the example, the system comprises four CSCIs (CSCI 1 through CSCI 4). Portions of the source elements from CSCI 2 are required by CSCI 1 in order to transform its source elements into its executables ❶. Portions of the source elements of CSCI 2 are required by CSCI 3 in order to transform its source elements into its executables ❷. CSCI 2 does not need any source elements from CSCI 1 or 3 to transform its source elements into its executables ❸. CSCI 4 does not need any source elements from any other CSCI to transform its source elements into its executables nor does any other CSCI use any of CSCI 4's source elements to transform its source elements into its executables ❹. This example results in two software product specifications: one that includes CSCIs 1, 2, and 3 ❺ and one that contains CSCI 4 ❻.

The SPS contains or references (a) the source-level elements for the grouping of related CSCIs, (b) their associated transformation procedures, and (c) the executables (including data) created by those transformation procedures that comprise the software product. Although not a part of the software product, the SPS also references the "as built" set of documents that define (a) the requirements met by the software product, (b) the design description of the software product, (c) the software-related manuals, (d) the hardware associated with the software product, and (e) the system architecture documents, namely, the SSDD and the OCD. The documents referenced by the SPS that are not a part of the product description are needed in order to describe fully the context within which the software product must exist.

The SPS also contains or references the relationships between the above information and the specific adaptations required for every instantiation of the software product and the related information not included as part of the software product. For example, if elements of the software product (e.g., executables) are installed at multiple user sites where each site has a slightly different set of hardware, then the specific adaptations of the software product required to execute in each user site are included in the SPS. In other words, the changes required to the information contained in the paragraph above that are needed to create the adaptation for each installation are contained in the SPS.

The SPS also defines the means by which each element of the software product is unambiguously verified to be a configured element for that product for each instantiation of the product. For example, if executable elements of the software product are installed in multiple user sites and each executable requires adaptation because the hardware is different, then a checksum associated with each executable could be the means to ensure that the correct adaptation has been installed in the correct site and that each site contains a valid copy of their appropriate software product elements.

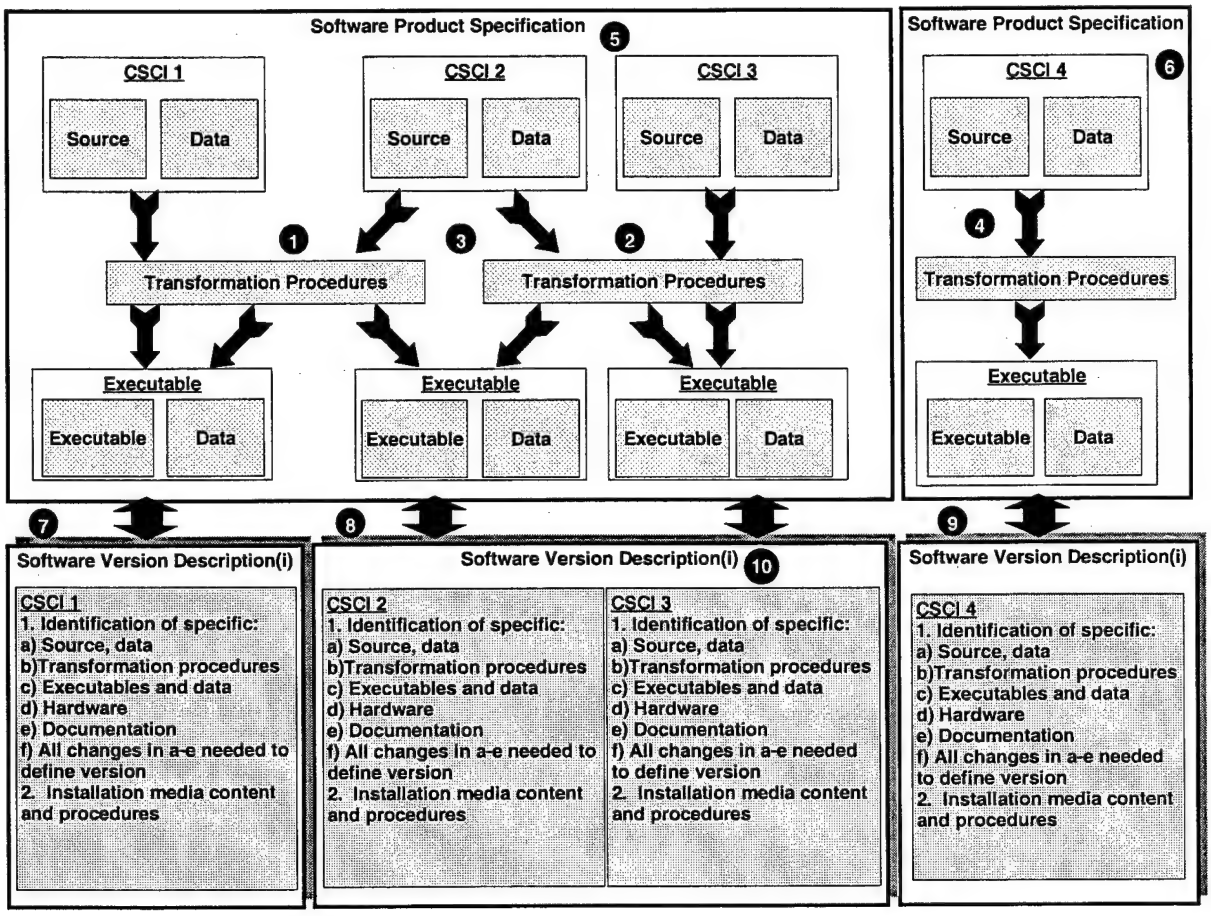


Figure 21. Software product specification/software version description relationship.

When an automated means is required to verify unambiguously that an element of the software product is a configured element (e.g., a file compare program or a checksum program is used to check automatically the correctness of a given adaptation against a configured description), that automation process is included in the Software Maintenance Manual. The method for creating the installation media for elements of a software product that represent an instantiation of the product also are contained in the Software Maintenance Manual. The description of the instantiation of the software product and the installation procedures for the software product are contained in one or more Software Version Description documents.

4.3.3 Software Version Description

The Software Version Description (SVD) is used to release, track, and control software product versions (instantiations) and their relationship to hardware versions, and system, software, and hardware documentation versions required by that software product (see Figure 21). The term "version" applies to the initial release of the software product, to a subsequent release of that same software product, or to one of multiple forms of that same software product released at approximately the same time (for example, to different operational or maintenance sites). The term "product element" applies to any specific source-level element, transformation procedure, or executable that is part of a software product.

When a software product is modified, an SVD document is used to define and describe the new version of the software product and all changes made to create the software product for each specific instantiation of that product. For example, elements of the new product (e.g., executables) may be delivered to multiple sites where each site has a slightly different hardware configuration that requires software adaptation. Each of the adaptations is included in the SPS in order to have a completely defined software product. Then, each SVD captures the specific elements of the SPS that are to be part of the specific instantiation of the product at each site and references all of the remaining elements of the software product to ensure that the SVD completely describes the software product.

The following discussion references Figure 21. Where appropriate to the discussion, associations between the text and different portions of the figure will be made using corresponding numbers in each.

In the example, the two software products ❺ and ❻ are to be installed at a number of sites (e.g., a maintenance site and multiple operational ground stations). Because of the system design, CSCI 1 will be installed in one set of equipment, CSCI 2 and CSCI 3 will be installed in another set of equipment, and CSCI 4 will be installed in a third set of equipment. This design most efficiently translates into the use of three SVDs to define the software versions to be installed at each site ❼, ❽, and ❾. SVDs for only one site are shown in the figure. The index "(i)" next to the name and the box shading indicate that the set of SVDs is repeated for each site ❿.

Each SVD used to identify a new version of a software product to be instantiated (e.g., at a specific site) performs two functions. The first is to identify (directly or by reference) each element of the product that actually has changed for the new instantiation, the changes made to each element that create the new product, and the instructions needed to actually perform the instantiation (e.g., installation of executables at a specific site). For this function, the SVD provides a complete identification of all software product elements (e.g., executables and data) and their changes that are to be included in the instantiation. The complete identification of the software product elements includes an unambiguous description (e.g., checksums of executable files) for each software product element included in the instantiation.

The second function is to identify (directly or by reference) the total set of elements that constitute the new product so that each SVD provides a complete standalone description of the total product, not just those elements of the product to be instantiated.

The STrP (see Subsection 2.6.2) is used to plan for the transition of software from location/use A to location/use B. The SPS is used to define completely the total software product, while the SVD defines (directly or by reference) the specific software product elements that are to be transitioned. The media used to install the software product elements described by the SVD are created using procedures documented in the Software Maintenance Manual. The integration and verification procedures associated with the software product elements described by the SVD are contained in the Software Maintenance Manual (or the Software Product Specification if no Software Maintenance Manual is to be delivered).

4.3.4 Computer Programming Manual

The Computer Programming Manual provides the information needed to program a given computer and its peripheral equipment (if appropriate). The manual focuses on the computer itself, not on particular software that will run on the computer. It applies to the programming of newly developed computers or special-purpose computer systems that do not have appropriate commercial manuals.

4.3.5 Firmware Support Manual

A system may contain many firmware devices. Each distinct device type of the system is included in the Firmware Support Manual (FSM). This includes Read Only Memory (ROM), Programmable ROM (PROM), and Erasable PROM or EPROM.

For each firmware device in the system, the FSM provides the information needed to erase the firmware device (if appropriate), to install the software to be contained within the firmware device into that firmware device, to verify that the installation was successful, and to mark the loaded device.

The front end of the firmware process consists of the development or modification of the software contained in the firmware device. This manual does not apply to the development/modification and verification of the software itself. The information needed to develop/modify and verify the software in the firmware device is contained in the Software Maintenance Manual and the appropriate SPS.

The back end of the firmware process consists of specific verification processes needed to certify that the device or the hardware assembly containing the device is ready for use in the operational system. This manual does not apply to the post-installation hardware verification needed for final certification prior to operational use. The final hardware verification used for certification of the device is contained in the hardware Tech Orders a) for that specific device or b) for an appropriate element of hardware that contains the device at a higher level of assembly.

The FSM contains pointers to the appropriate SVDs and SPSs for the software contained in each device. It also contains pointers to the appropriate system hardware drawings and Hardware Product Specifications that contain each device.

4.3.6 Hardware Product Specifications

The Hardware Product Specification defines (either directly or by reference) the "as built" hardware drawings that constitute the hardware product. This consists generally of HWCIs and their associated lower-level drawings for each hardware assembly in the HWCI that is not considered a piece part. Each drawing (except the highest and the lowest) will have a pointer(s) to the appropriate higher-level assembly(ies) and lower-level assembly(ies). Each drawing will have a pointer to each firmware device that it contains at its level of assembly, but not for those devices included on lower-level assemblies. Each drawing will be able to be related to the specific tech order(s) applicable to this drawing level.

The Hardware Product Specification defines the drawings [and revision numbers/engineering change proposals (ECPs)] for all instantiations of that HWCI. The Software Product Specification references the Hardware Product Specification to define the hardware requirements associated with a specific software product. The Hardware Product Specification references the specific Tech Order versions that are applicable to the specific hardware product.

4.4 System Operation and Maintenance

4.4.1 Computer Operator Manuals

The Computer Operator Manual (COM) provides the information needed to operate a given computer and its peripheral equipment. The manual focuses on the computer itself, not on particular software that will run on the computer. It applies to newly developed computers or special purpose computer systems that do not have appropriate commercial manuals that describe the computer operator position. For newly developed computers or special purpose computer systems, this manual really should be a positional handbook for the computer operator. In some cases, Tech Orders are used to document the information normally contained in the COM.

4.4.2 System Hardware Drawing Package

The System Hardware Drawing Package provides the "as built" drawing tree for the entire system. This package provides (either directly or by reference) the drawings for all of the hardware in the system, their

hierarchical and peer relationships, and the changes (e.g., ECPs) incorporated in the hardware. The Hardware Product Specification may reference elements of this package to describe the specific drawings associated with a specific hardware product that relates to a specific software product.

4.4.3 System Integrated Diagnostics Document

The System Integrated Diagnostics document provides both operations personnel and maintenance (repair) personnel with a system-level view of the diagnostics available for restoring system operations following a fault (system restoral) and for the ultimate diagnosis and repair of that fault (system repair). The other documents included in the figure and the existing operations and maintenance documents treat hardware and software as independent elements within the system and do not provide information that will allow an operator or maintainer to isolate a particular system fault to an individual hardware or software component from the top down. This document transcends specific positions and provides a coordinated mechanism for fault detection, isolation, and localization that allows system faults to be allocated to specific positions or organizations for further action (both restoral and repair actions). Further hardware actions are included in the appropriate Positional Handbooks, Software User Manuals (if the hardware problem is related to computer resources), or Tech Orders associated with the specific problem. Further software actions are included in the appropriate Positional Handbooks, Software Users Manuals, Software Maintenance Manual, or Tech Orders (if the software problem relates to computer resources and Tech Orders are used instead of Computer Operator Manuals).

4.4.4 Hardware Tech Orders

Hardware Tech Orders provide the information needed to administer, operate, and maintain the equipment contained within a specific system. This includes maintenance as well as operational equipment. The Tech Orders for a specific system must relate to the specific version(s) of equipment to which they apply.

Appendix A: Verification Methods

For the purpose of this report, wherever verification methods are mentioned, the following definitions are assumed:

a. **Inspection:** A method used to determine characteristics by inspecting engineering documentation produced during product development (including both hardware and software documentation) or by inspection of the product itself to verify conformance with specified requirements. Inspection generally is nondestructive and consists of visual inspections or simple measurements without the use of precision measurement equipment.

b. **Analysis:** A method used to verify requirements by determining qualitative and quantitative properties and performance by studying and examining engineering drawings, software and hardware flow diagrams, software and hardware specifications, and other hardware and software documentation (e.g., COTS vendor documentation), or by performing modeling, simulation, and/or calculations and analyzing the results. Similarity analysis is used in lieu of tests or demonstrations when it can be shown that an item is similar to or identical in design to another item that has been certified previously to equivalent or more stringent criteria.

c. **Demonstration:** A method used to verify requirements by exercising or operating the system or a part of the system in which instrumentation or special test equipment is not required beyond that inherently provided in the system being verified. In the demonstration method, sufficient data for requirements verification can be obtained by observing functional operation of the system or a part of the system. When this verification method generates data that are recorded by inherent instrumentation, inherent test equipment, or operational procedures, any analysis that must be performed using the data collected during the demonstration is an integral part of this method and should not be confused with the analysis method of verification described above.

d. **Test:** A method used to verify requirements by exercising or operating the system or a part of the system using instrumentation (hardware and/or software) or special test equipment that is not an integral part of the system being verified. The test method by its nature generates data, which are recorded by the instrumentation, test equipment, or procedures. Analysis or review is performed on the data derived from the testing. This analysis, as described here, is an integral part of this method and should not be confused with the analysis method of verification described above.

Appendix B: Interface Association Traceability Information Detail

This appendix provides a more in-depth discussion of the interface association traceability information required in the Traceability/Accountability Repository than that found in Subsection 3.9.

The Traceability/Accountability Repository must contain traceability related to the bi-directional association of interfaces with each other in terms of both the evolving system requirements and the evolving design. The interface association traceability must be organized in such a manner that meaningful technical analysis about the interface can be performed; for example, answers to such questions as "How do the physical and functional requirements of the interface itself relate to the amount and type of messages that flow though the interface?," or "How do changes to requirements about a specific message that traverses the interface impact the interface and its requirements?." Maintaining the interface traceability is as critical to the successful control and management of the program as maintaining requirements traceability. In order to discuss effectively the needs and complexity of interface association traceability, there needs to be a consistent description of what interfaces are and how they are structured and decomposed. The next subsection defines the elements of such a consistent description. This model is assumed whenever interfaces are discussed. The description and definitions will be followed by a simplified example showing the characteristics and complexity of the interface association traceability required in the management of a program.

Interface Structure and Decomposition

Interface structure and decomposition have been the focus of much research as distributed computing and networks have proliferated. This research has resulted in a number of interface protocol standards that provide a consistent method for describing how interfaces should be implemented to ensure an open system architecture. The International Standards Organization has developed a seven-layer Open Systems Interconnection Reference Model (OSIRM). This reference model focuses on the way to develop and implement interfaces but does not provide a sufficiently robust descriptive framework for specifying interfaces and their relationships. In particular, this model does not include process-to-process interfaces.

In order to provide a complete framework for interface description at the specification level, the OSIRM model has been generalized into a Layered Interface Definition (LID) model. The LID has been defined and documented in Aerospace Technical Operating Reports (Refs. 3, 4). The LID model is a standardized approach for describing, categorizing, and defining the requirements of an interface and its functions. The LID model has been chosen for use in this report as a framework for describing interfaces. Although this is not the only method for describing interfaces and their relationships, it provides the generality needed to convey the concepts important to this report. The LID takes the seven-layer OSIRM interface model and maps it to its four-layered model. The relationship of the layers is shown in Figure 22. This figure also shows the relationship of the LID layers to the interfaces between two distributed processes. The LID model also allows for the possibility of interface information that is not layered.

The LID Interprocess layer contains interface details describing requirements among end-user procedures that are necessary for the processing of information used by those procedures, exclusive of data interpretation (interpretation layer) and data transfer (path and link layer) relationships. The Interprocess layer does not correspond to any OSIRM layer and lies above all other LID layers. This layer includes interface requirements for both software and hardware process interfaces as well as human-computer semiautomatic process interfaces.

The LID Interpretation layer contains interface details describing requirements for the synchronization, syntax, and semantics of data transfer across an interface boundary. It addresses the end-to-end

functional requirements for the interface. This layer includes interface requirements for data transfer functions that are visible to the end-user or involve the end-user, such as semiautomatic applications software as well as communications.

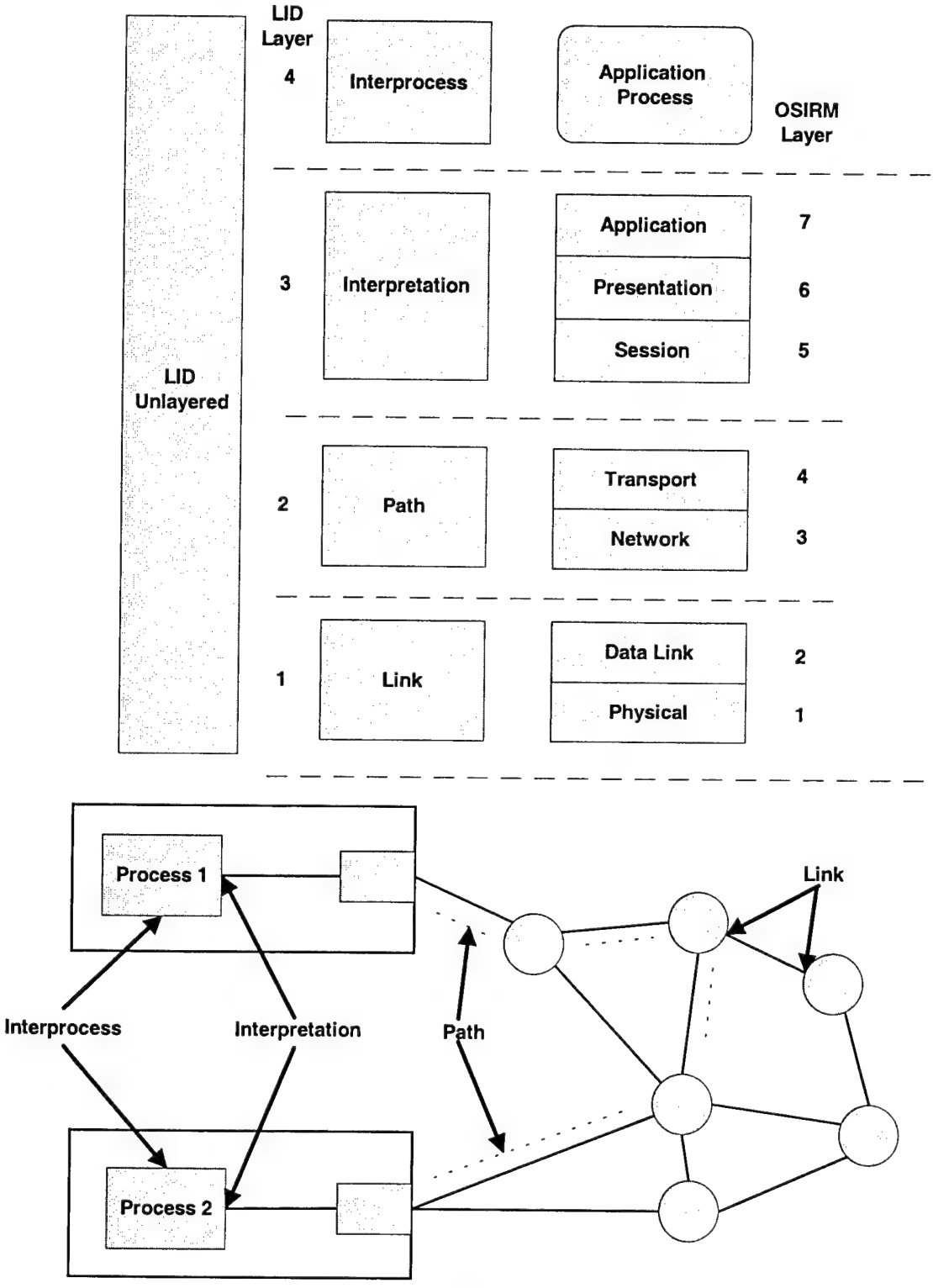


Figure 22. Layered interface definition model.

The LID Path layer contains interface details describing requirements that relate to the path of the data from one system to another. The interface defined in this layer must operate successfully over a path without interfering with other users of the interconnected media. This layer includes interface requirements for the issues of priority, reliability, routing, and quality of service.

The LID Link layer contains interface details describing the requirements of the individual links between adjacent items. It defines the portion of the interface dealing with link control and media definition. This layer includes the interface requirements for the need for services, for speed of service, flow control, and error control.

The Unlayered portion of the LID contains interface details describing requirements directly related to supporting the end processes such as security requirements, or conditions or dependencies that do not relate to any layer.

The following example is presented in order to provide a basic understanding of the LID layer concepts. In this example, there are two persons at two separate locations. One person develops a document on a word processor and sends it to the other via modems and telephone lines. The other person then receives the document and reads it.

Interprocess Layer: The persons generating and reading the documents at both locations are the end users. The reasons for creating and reading the documents constitute the requirements for the Interprocess Layer.

Interpretation Layer: The word processing software that is needed to create the document to be sent and to read the document once received must be compatible. The document data formats and content structure constitute the requirements for the Interpretation Layer.

Path Layer: The document must be addressed and prepared for transmission from the originating site to the receiving site. The method of transfer, the communications software required to transfer data via the modems, the telephone numbers (addressing), and other routing data would constitute the requirements for the Path Layer.

Link Layer: The document must have a medium for transmission between sites. If the interface boundary is at the point where the modem output connects to the telephone line, the electrical characteristics of the signal and the mechanical characteristics of the connection constitute the requirements for the Link Layer.

Unlayered Portion: The document contains sensitive information that must be protected. The encryption level and method constitute the requirements of the Unlayered Portion of the model.

Interface Association Traceability Simplified Example

In order to illustrate the importance and complexity of interface association accountability, a simplified example from a typical satellite system is shown in Figure 23. This simplified example includes only software-to-software interfaces. The added complexity of software-to-hardware and hardware-to-hardware interfaces, their relationship to software-to-software interfaces, and the required interface traceability tracking for them is not included but is similar to the description of software-to-software interfaces given below.

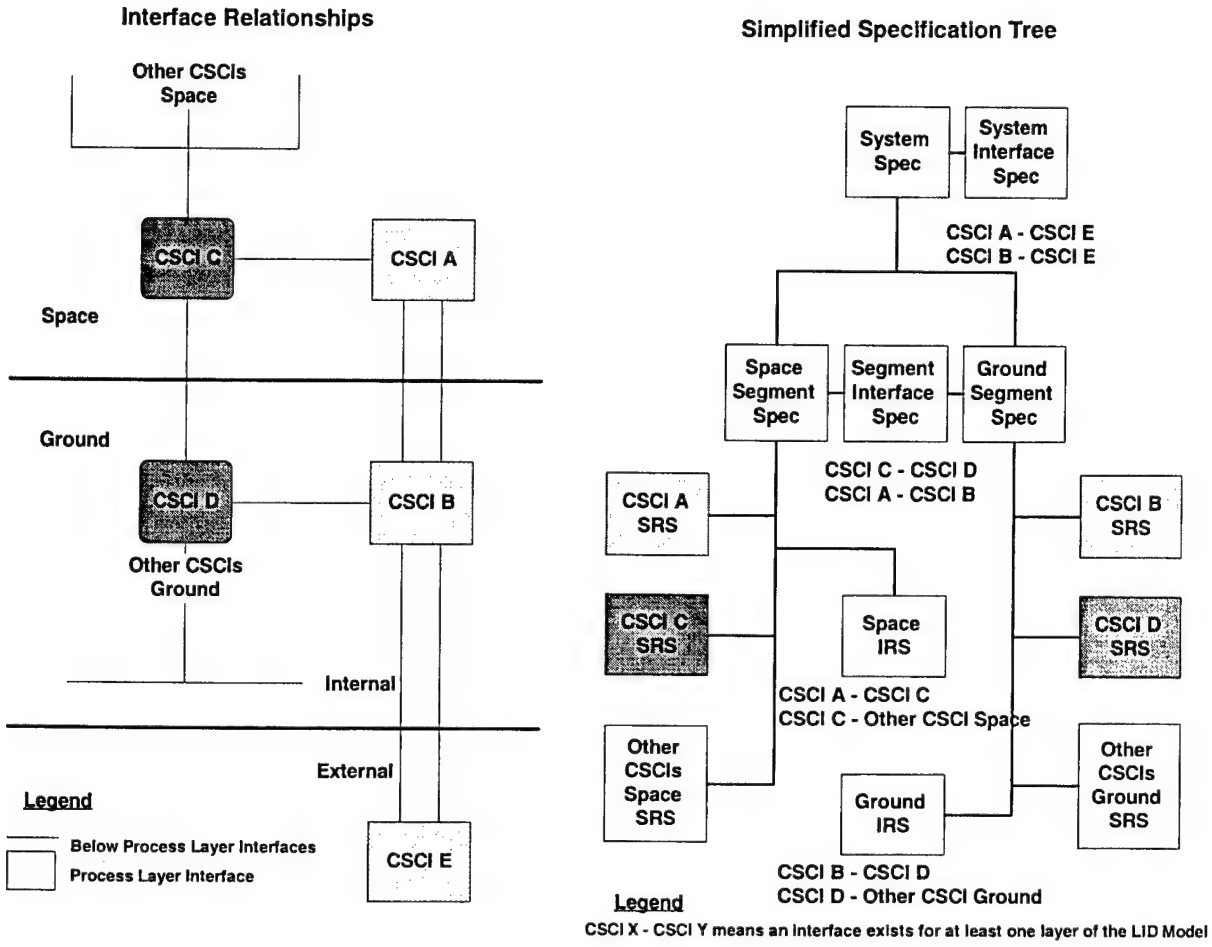


Figure 23. Interface association tracking.

In this example, it is assumed that:

a. The system consists of two segments (Space and Ground).

b. There are five CSCIs of primary interest to the example, CSCIs A through E (where CSCI E is external to the system).

c. There is more than one other CSCI in space and on the ground that also interact with two of the CSCIs of primary interest (CSCIs C and D). In this example, the other CSCIs collectively are called "Other CSCIs Space" and "Other CSCIs Ground" in the figure.

d. CSCI A and CSCI B interface in that they have interrelated requirements for the processing of data (e.g., CSCI A collects and creates related event observations, while CSCI B takes the related event observations and creates two- and three-dimensional event tracks). This interface is shown as a double line between CSCI A and CSCI B in the figure.

e. CSCI B has an interface with the external CSCI E via a system external communications link[29] (e.g., CSCI B formats and distributes the resultant event tracks to external users in user-defined formats for processing).

[29] For simplicity, any interprocessor communication software needed by the various CSCIs to communicate with the various space data buses and ground LANs is ignored in the example but must be included for any "real" system.

f. CSCI A and CSCI E interface in that they have interrelated requirements for the passing of raw unprocessed data (e.g., CSCI A collects unprocessed event observations and provides them to CSCI E for non-real-time processing). Although CSCI A connects to external CSCI E through CSCI B, it is assumed that CSCI B does not have processing responsibility for the data beyond the necessity for packaging the information for CSCI E.

g. CSCI A and C interface via one data bus (e.g., the payload bus).

h. The "Other CSCIs Space" and CSCI C interface via another data bus (e.g., the spacecraft bus).

i. CSCI C and D interface via an intersegment space-to-ground communications link.

j. CSCI D and B interface via one LAN (e.g., the payload LAN).

k. CSCI D and the "Other CSCIs Ground" interface via another LAN (e.g., the health and status LAN).

l. Although in the general case at least some of the "Other CSCIs Space" and "Other CSCIs Ground" would have process-to-process interfaces, for the purposes of this example, the only interfaces considered at the process level (the Interprocess Layer in the LID model) are between CSCI A and CSCI B, between CSCI A and CSCI E, and between CSCI B and CSCI E. All other interfaces between CSCIs are at lower layers in the LID model (i.e., either Interpretation, Path, or Link).

Using the specification and design documentation philosophy discussed in the example in Subsection 3.3, this example's simplified specification tree, as shown in Figure 23, consists of a system specification, two segment specifications (space and ground), and several B-level specifications including SRSs for CSCI A, C, and the "Other CSCIs Space" (a collective reference to one or more other space CSCI SRSs) in the space segment and SRSs for CSCIs B, D, and the "Other CSCIs Ground" (a collective reference to one or more other ground CSCI SRSs) in the ground segment. There also is a system interface specification for the external interface between CSCI B and CSCI E and between CSCI A and CSCI E, a segment interface specification for the intersegment interface between CSCI C and CSCI D, and the intersegment interface between CSCI A and CSCI B. There would be at least one IRS for the appropriate interfaces among CSCIs A, C, and the "Other CSCIs Space" in the space segment and at least one IRS for the appropriate interfaces among CSCIs B, D, and the "Other CSCIs Ground" in the ground segment.

The SRSs in this example would reference the interface requirements in the appropriate interface specifications as shown in Table 1 below.

Table 1. SRS Software-to-Software Interface References

Space Segment CSCIs Interface References	Ground Segment CSCIs Interface References
CSCI A SRS A1) System Interface Spec for CSCI A - CSCI E A2) Segment Interface Spec for CSCI A - CSCI B A3) Space IRS Spec for CSCI A - CSCI C	CSCI B SRS B1) System Interface Spec for CSCI B - CSCI E B2) Segment Interface Spec for CSCI A - CSCI B B3) Ground IRS Spec for CSCI B - CSCI D
CSCI C SRS C1) Segment Interface Spec for CSCI C - CSCI D C2) Space IRS Spec for CSCI A - CSCI C C3) Space IRS Spec for CSCI C - Other CSCIs Space	CSCI D SRS D1) Segment Interface Spec for CSCI C - CSCI D D2) Ground IRS Spec for CSCI B - CSCI D D3) Ground IRS Spec for CSCI D - Other CSCIs Ground

Using the LID model for this example would result in the software-to-software interface associations being structured as shown in Figure 24. The identifiers on the connections correspond to the labels in Table 1. As shown in this example, many common lower-layer interface associations are used to effect the interfaces at the Interprocess layer. For example, interface C1 and D1 is a common element of Interface 1, Interface 3, and Interface 4. This is the space-ground link in this example. The interface B1 is common to Interface 2 and Interface 3. This is the external interface in this example. The Traceability/Accountability Repository must maintain the horizontal associations at each layer in the interface structure (e.g., CSCI A - CSCI C - CSCI D - CSCI B in Interface 1) as well as the vertical associations for the common layer elements (e.g., the CSCI C - CSCI D interface layer is included in Interface 1, Interface 3, and Interface 4). The maintenance of these horizontal and vertical associations of the interface layers provides the basic information necessary to answer questions such as "Is there sufficient capacity in the space-ground link (CSCI C - CSCI D) to handle all of the required data traffic?" or "Since the external interface has changed requirements (CSCI E), what impact does that have on other system CSCIs (CSCI B - CSCI E, CSCI A - CSCI E, CSCI B - CSCI D, CSCI C - CSCI D, CSCI A - CSCI C)?"

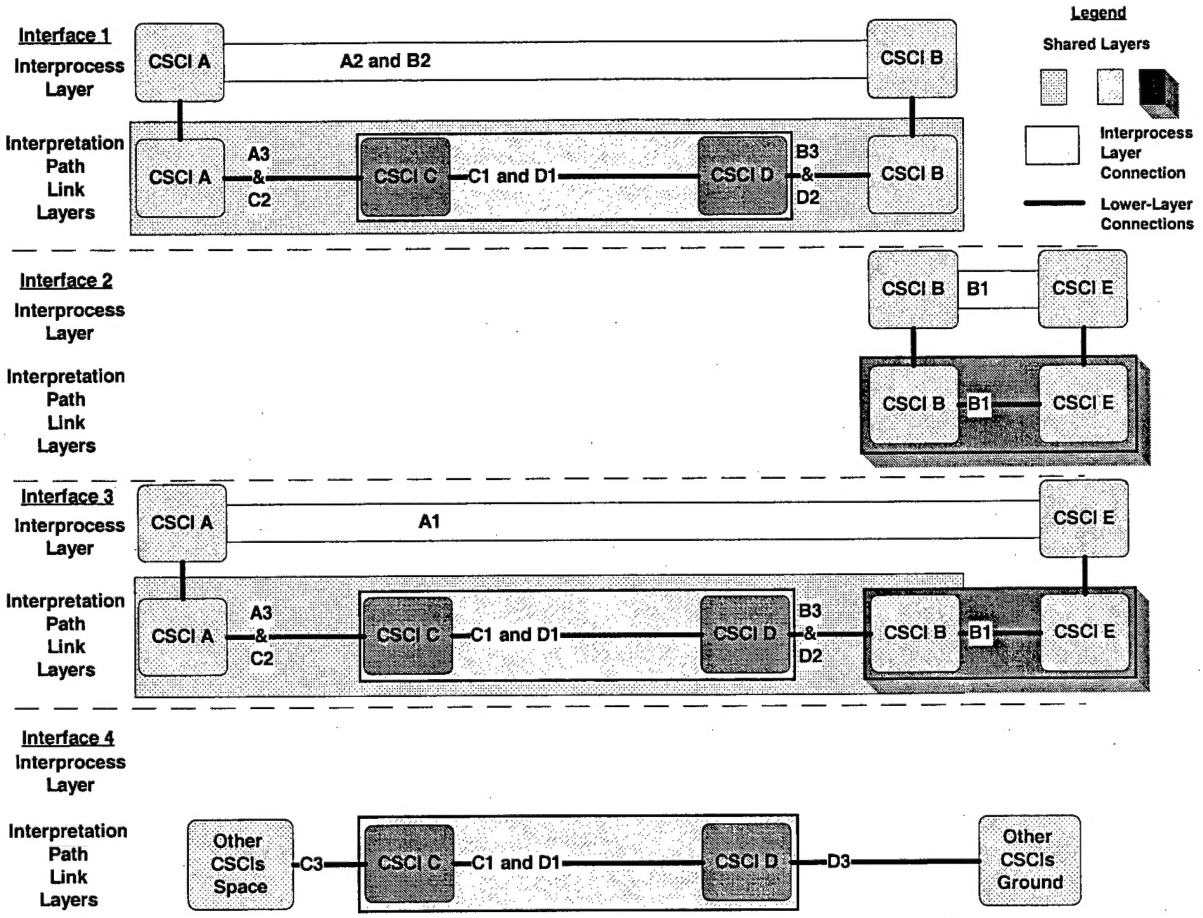


Figure 24. Layered communications model interface associations.

References

1. "Software Development and Documentation," MIL-STD-498, 5 December 1994.

2. "Standard for Information Technology - Software Life Cycle Processes - Software Development: Acquirer-Supplier Agreement," EIA/IEEE J-STD-016-1995, 21 September 1995.

3. Lanzinger, D. J. and R. E. Berri with A. Hoheb, "Handbook for Interface Development and Support Engineering (IDSE) for the Air Force Satellite Control Network (AFSCN)," Report No. TOR-0091(6488-04)-1, Reissue A, September 1993.

4. Lanzinger, D. J. and R. E. Berri with A. Hoheb, "Handbook for Interface Development and Support Engineering (IDSE) for the Air Force Satellite Control Network (AFSCN)," Report No. TOR-0091(6488-04)-2, Reissue B, September 1993.